Always Online Infrastructure (AOI)

Approved Version 1.0 – 04 April 2017

Open Mobile Alliance

OMA-ER-AOI-V1_0-20170404-A

Contents

Figures

Tables

1. Scope

This Enabler Release (ER) document is a combined document of requirements, architecture and technical specification for Always Online Infrastructure Enabler. The AOI Enabler is expected to provide functions of application/service management, connection management, interfaces between AOI Client and AOI Server, interfaces to client applications and server applications and bandwidth optimization.

2. References

2.1 Normative References

[JSON-RPC] JSON-RPC 2.0 URL:http://www.jsonrpc.org/specification.

[OMA-PUSH] "OMA Push 2.3" Open Mobile Alliance™. OMA-ERP-Push-V2_3 URL:http://www.openmobilealliance.org/

[OMA-PUSH-REST] "OMA Push 2.3" Open Mobile Alliance™.OMA-ERP-REST_NetAPI_NotificationChannel-V1_0-20130730-C URL:http://www.openmobilealliance.org/

[OSE] "OMA Service Environment", Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC4234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. October 2005, URL:http://www.ietf.org/rfc/rfc4234.txt

[RFC4627] "The application/json Media Type for JavaScript Object Notation (JSON)", Crockford, D., July 2006.

URL: http://www.ietf.org/rfc/rfc4627.txt

[RFC6455] "The WebSocket Protocol", I. Fette and A. Melnikov, December 2011, URL: http://www.ietf.org/rfc/rfc6455.txt

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL:http://www.openmobilealliance.org/

[SpamReport] "SpamReport", Version 1.0, Open Mobile Alliance™, OMA-TS-SpamRep-V1_0-20120619-A, URL:http://www.openmobilealliance.org/

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_9, URL:http://www.openmobilealliance.org/

[W3C_WebSock] "The WebSocket API", W3C Candidate Recommendation 20 September 2012, Ian Hickson, ed., URL: http://www.w3.org/TR/websockets/

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Active AOI Client	An AOI Client that is installed in device and registered on an AOI Server successfully.
AOI Client	A client (see [OMA-DICT]) that acts as the receiver of a AOI service in order to realize Always-Online connectivity. The AOI Client does not render (e.g. display, play etc) application specific content, but provide AOI functions for the Client Application.
AOI Server	A Server (see [OMA-DICT]) capable of communicating with any AOI Client over Shared AOI Connectivity.
AOI Subscription Information	Information which is registered and subscribed to the AOI Enabler and uniquely identifies the Client Application across all devices.
Client Application	An application (see [OMA-DICT]) which can be native or a web application that uses or triggers the AOI Service using the AOI Client. Examples of Client Applications can be social networking applications, VoIP and instant messenger applications which reside on the mobile terminal.
Client Application Instance	A single installation of an Client Application in a device.
Non-Active AOI Client	An AOI Client that is installed in device, but is not registered on any AOI Server.
Server Application	An application (see [OMA-DICT]) that uses or triggers the AOI Service using the AOI Server. Examples of Server Applications can be social networking applications, VoIP and instant messenger applications which reside on the network side.
Server Application Provider	A provider which provides Server Application. Single provider can provide multiple Server Applications.
Shared AOI Connectivity	A connectivity maintained by AOI client and AOI Server over connection-less bearer (e.g., SMS, OMA Push) or connection-oriented bearer (e.g., HTTP, TCP) where multiple Client Applications share the connectivity for the purpose of interaction with Server Application.

3.3 Abbreviations

AOI	Always Online Infrastructure
IMSI	International Mobile Subscriber Identity
MNO	Mobile Network Operator
MSISDN	Mobile Station International Subscriber Directory Number
NAT	Network Address Translation
OMA	Open Mobile Alliance
OMC	Operations and Maintenance Center
OTA	Over The Air
PAP	Push Access Protocol
REST	Representational State Transfer
SIM	Subscriber Identity Module

SMS	Short Messaging Service
TCP	Transport Control Protocol
UDP	User Datagram Protocol

4. Introduction

The AOI Enabler defines a notification framework that on one hand addresses the needs of application developers, and on the other hand minimizes the impact on the network and improves devices' battery life, and as such improving the overall experience of final users.

4.1 Challenges

Two main alternatives can be used by clients to asynchronously retrieve information from a server:

- Polling: periodically querying the server to check whether there is new information, what is not efficient as it requires connections to be made even if there is no new information to be retrieved
- Push: The server sends the information to the client as new information becomes available

In order to build a real Push service the server needs to have the means to reach the client. However this is not always possible when the device is connected using a mobile data connection and the server is placed outside the MNO's network, i.e. in the Internet, as mobile devices are usually placed behind a firewall/NAT and are not assigned a permanent public IP address accessible from the internet. Mobile devices are assigned instead a private IP address valid in the realm of the MNO's network, which is temporarily mapped with a public IP address at the Firewall/NAT. This temporary association is released once the connection exceeds an inactivity threshold set by the MNO. As public IP addresses are a scarce resource, MNOs usually release these associations after relatively short inactivity periods (in the range of minutes).

Some notification systems circumvent the previous issue by establishing a connection between the client and the server and keeping some activity in that connection (i.e. keep-alive messages) which forces the Firewall/NAT to keep the association between public and private IP addresses even if there is no actual data to be transferred from the server.

Sending keep-alive messages when there is no active data connection in place requires the mobile device to activate the data connection in order to send a few bytes and then release it again. These transitions from an idle radio state to active state constitute a huge signaling burden on mobile networks, as well as have a dramatic impact on devices' battery life. If different clients (e.g. applications) maintain such connections with different servers in an uncoordinated fashion the problem worsens, causing the so-called 'signaling storm'.

4.2 How to solve the problem

The problem for the case when Device is within the mobile network can be solved by relying on an entity that is placed within the MNO's network and thus can reach the mobile device without the need to have a public IP address permanently associated.

Different approaches may be used by this entity to allow the mobile device to remain in an idle radio state, in which it just listens to wake-up messages, the so-called 'paging messages':

- Mechanisms based on SMS, as for instance delivering notifications using OMA Push over SMS
- Mechanisms based on the submission by the server of an UDP/TCP packet to the private IP address of the device to make it wake up and establish a data connection. This mechanism just requires the mobile device to have a private IP address assigned (i.e. an active PDP context), which does not imply consuming any data resources.

These approaches are compatible with a scenario in which the notification server (i.e. AOI Server) is placed outside of the MNO's network, in which case an auxiliary server placed within MNO's network may support the notification server in performing previously mentioned mechanisms.

In the case when device is within a Wi-Fi or another non mobile network with Firewall/NAT and AOI Server is outside this network, Client-initiated TCP notification channel setup between AOI Client and AOI Server is an alternative in addition to SMS. In this case Auxiliary server is not required.

4.3 Version 1.0

This release of AOI Enabler defines a notification platform that covers at least the following functions:

- Notification channel management, i.e. notification channel setup/activation and channel termination/deactivation
- Connectivity management, e.g. notification channel maintenance
- Handling of notification information, e.g. assigning priority levels for notifications
- Delivery of notification
- Reporting notification statistics information

5. Requirements (Normative)

This section captures the requirements for AOI v1.0.

5.1 High-Level Functional Requirements

This section contains the High Level requirements for AOI v1.0.

Label	Description	Release
AOI-HLF-001	The AOI Enabler SHALL allow AOI Client to register to/ de-register from AOI Server.	1.0
AOI-HLF-002	The AOI Enabler SHOULD allow Client Application to register to/ de-register from AOI Client.	1.0
AOI-HLF-003	The AOI Enabler SHOULD allow Server Application to register to/ de-register from AOI Server.	1.0
AOI-HLF-004	The notification mechanism over Shared AOI Connectivity SHOULD support multiple bearers (e.g. use SMS when IP connectivity is not available)	1.0
AOI-HLF-005	The AOI Enabler SHALL provide a dedicated notification mechanism over Shared AOI Connectivity in lightweight manner.	1.0
AOI-HLF-006	The AOI Enabler SHALL support an adaptable communication mechanism between the AOI Client and AOI Server depending on the policy and capability negotiation between AOI Client and AOI Server. **Informational Note**: adaptable frequency of communication between the AOI Client and AOI Server based on network condition and application requirements.	1.0
AOI-HLF-007	The AOI Enabler SHALL support the AOI Client to send lightweight messages to AOI Server using a Shared AOI Connectivity.	1.0
AOI-HLF-008	The underlying bearer of the Shared AOI Connectivity in place at a specific point of time between the AOI Server and a specific AOI Client SHALL be transparent to the AOI Applications, i.e. Client Application and Server Application	1.0
AOI-HLF-009	The AOI Enabler SHALL be able to detect the changes of subscriber information (e.g. IMSI, MSISDN).	1.0
AOI-HLF-010	The AOI Enabler SHALL be able to report the detected changes of subscriber information (e.g. IMSI, MSISDN). **Informational Note**: Reporting the changes does not imply reporting the information	1.0
AOI-HLF-011	The AOI Enabler SHALL be able to support AOI service even if the end user moves out of home network of the Network Operator originally providing the AOI Service.	DELETE
AOI-HLF-012	The AOI Enabler SHALL support size limitation of messages.	1.0
AOI-HLF-013	The AOI Enabler SHALL be able to enforce the policies and values of size limitation of messages	1.0
AOI-HLF-014	The AOI Enabler SHALL be able to manage (e.g., add, remove, update) application information stored in the AOI Client and/or Server (i.e., Client Application and/or Server Application related information such as name of application, address/information used for notification delivery).	1.0
AOI-HLF-015	The AOI Enabler SHOULD support a backoff mechanism in order to avoid message flooding potential risk. **Informational Note**: as an example, an exponential backoff mechanism may do that a AOI Client waits before reattempting to connect to the AOI Server for a time which exponentially increases compared to the previous waiting time value, in case the connection attempt is failed due to AOI Server failure.	Future Release
AOI-HLF-016	The AOI Enabler SHOULD make available statistics information of the AOI Server or AOI Client (e.g. connect/disconnect activities, connection history, time period, number of notifications, number of kB sent or received, upload/download speed) .	1.0

AOI-HLF-017	The AOI Enabler SHOULD be able to collect statistics information (e.g. connect/disconnect activities, connection history, time period, number of notifications, number of kB sent or received, upload/download speed).	1.0
AOI-HLF-018	The AOI Enabler SHALL support the communication between a Server Application and Client Application through respective AOI Server(s) and AOI Clients to which Server/Client Applications are connected to.	DELETE

5.2 Application State Definition in AOI

Label	Description	Release
AOI-ASD-001	The AOI Enabler SHALL be able to manage (e.g. query, update) different relational status information (e.g. registered, de-registered, online, offline, installed, uninstalled) related to Client Application. **Informational Note**: status of Client Application is not an application specific status but is related to the AOI Enabler, such as registration status of the Client Application to the AOI Enablers.	1.0

5.3 Notification Management

Label	Description	Release
AOI-NOT-001	The AOI Enabler SHALL allow AOI Client to subscribe/un-subscribe the notification from AOI Server according to Client Application registration and de-registration.	1.0
AOI-NOT-002	The AOI Enabler SHOULD allow the end users to activate/deactivate notifications for all applications or per specific application.	1.0
AOI-NOT-003	AOI Enabler SHOULD support handling metadata in notification messages based on the local/regional/language and time zone specifics (e.g. an error code sent by the AOI Server to the AOI Client may be handled at the AOI Client based on the local language and time zone).	Future Release
AOI-NOT-004	The AOI Enabler SHALL make available the necessary information to Server Application so that notification can be delivered through AOI Server and AOI Client.	1.0
AOI-NOT-005	The AOI Enabler SHALL allow Server Application to send notifications to an instance of a Client Application installed in a specific device.	1.0
AOI-NOT-006	The AOI Enabler SHALL allow Server Application to send notifications to all instances of a Client Application that are associated to an end user of that application and that are installed in devices with AOI Clients registered in an AOI Server.	1.0
AOI-NOT-007	The AOI Enabler SHALL allow Server Application to send notifications to all instances of a Client Application associated to all end users of that application that are installed in devices with AOI Clients registered in an AOI Server.	1.0
AOI-NOT-008	The AOI Enabler SHALL allow Server Application to send notifications to a selected set of instances of a Client Application associated to a selected set of users of that application that are installed in devices with AOI Clients registered in an AOI Server.	1.0
AOI-NOT-009	The AOI Enabler SHALL be able to support the Server Application to check if a specific notification has been delivered to the AOI Client	1.0
AOI-NOT-010	The AOI Enabler SHALL allow the Server Application to request to be notified if one or multiple notifications have been delivered to the AOI Client.	1.0
AOI-NOT-011	The AOI Enabler SHOULD be able to support the Server Application to check if a specific notification has been delivered to the Client Application.	1.0
AOI-NOT-012	The AOI Enabler SHOULD allow the Server Application to request to be notified if one or multiple notifications have been delivered to the Client Application.	1.0
AOI-NOT-013	The AOI Enabler SHOULD allow the Server Application to check the delivery status of multiple notifications (i.e., if those notifications have been delivered to the AOI Client Application) in a single request and receive result in a single response.	Future Release
AOI-NOT-014	The AOI Client SHOULD be able to alert the end user on the reception of notification even if the Client Application is not running.	1.0

AOI-NOT-015	If the AOI-HLF-025 is supported, the AOI Client SHALL be able to delete the received notifications which have been delivered to the end user.	DELETE
AOI-NOT-016	The AOI Client SHALL be able to delete received notifications which have been delivered to the Client Application.	1.0
AOI-NOT-017	The AOI Enabler SHALL allow Server Application to set an expiry time within which the notification needs to reach the destination.	1.0
AOI-NOT-018	The AOI Server SHALL NOT deliver to the AOI Client a notification after the expiry time set by the Server Application.	1.0
AOI-NOT-019	The AOI Server SHALL provide APIs for Server Applications to use AOI services (such as sending notification messages, etc.).	1.0
AOI-NOT-020	The AOI Enabler SHOULD support end user information management.	Future Release
AOI-NOT-021	The AOI Server SHALL support mechanisms to control traffic for notification messages based on service provider policy, such as a threshold per application, selected applications or all applications.	1.0
AOI-NOT-022	The AOI Enabler SHALL support prioritized delivery of notifications based on service classes.	1.0
AOI-NOT-023	The AOI Enabler MAY support prioritized delivery of notifications based on end user classes.	1.0
AOI-NOT-024	The AOI Enabler SHALL allow Server Application to assign a prioritized service class to a specific notification.	1.0
AOI-NOT-025	The AOI Enabler SHOULD be able to retrieve network status information from external sources required for policy decision (e.g. interface with OMC system of Wireless Network to retrieve the system load indicator). **Informational Note**: the interface to the external sources is out of scope for the AOI Enabler.	Future Release
AOI-NOT-026	If network status information is available, the AOI Enabler SHALL support notification mechanism based on such information, e.g. which can buffer the notification messages (and push them when the network is idle) or reject the notification messages.	1.0
AOI-NOT-027	The AOI Enabler SHALL be able to store notifications for deferred delivery based on AOI service provider policy (e.g. the network is not available).	1.0
AOI-NOT-028	The AOI Enabler SHOULD be able, on request by the Server Application, to update or cancel the notification(s) sent by the Server Application but not yet delivered to Client Application.	Future Release
AOI-NOT-029	If AOI-NOT-028 is supported, the AOI Enabler SHALL be able to notify the Server Application if a request to update or cancel sent notification(s) has succeeded.	Future Release
AOI-NOT-030	The AOI Enabler MAY support compression of the notifications.	1.0
AOI-NOT-031	The AOI Client MAY report whether it supports compression to AOI Server. **Informational Note**: if the AOI Client has reported to AOI Server its support for compression, the AOI Enabler will take into consideration terminal capability.	1.0
AOI-NOT-032	The AOI Server SHALL be able to aggregate several notifications and transmit them at the same time to the AOI Client (e.g. a set of notifications not marked as real time may be sent together with the first real time notification that arrives).	1.0
AOI-NOT-033	The AOI Enabler SHALL be able to store notifications for deferred delivery based on end user's policy (e.g. end users do not want to be disturbed in some timeframe).	1.0
AOI-NOT-034	The AOI Enabler MAY provide mechanism to filter out malicious notification messages. **Information:** OMA SpamReport enabler does not prevent different filtering mechanisms for filtering spam [SpamReport]	1.0

AOI-HLF-035	The AOI Enabler SHALL be able to manage the delivery conditions and delivery status of notifications.	Future Release

5.4 Connection Management

Label	Description	Release
AOI-CM-001	The AOI Enabler SHALL be able to manage (e.g. create, maintain, close) the Shared AOI Connectivity between AOI Client and AOI Server.	1.0
AOI-CM-002	The Shared AOI Connectivity SHALL NOT be maintained unless at least one Client Application is registered with the AOI Client.	1.0
AOI-CM-003	The AOI Enabler SHALL be able to reduce the number and size of messages to keep Shared AOI Connectivity alive.	1.0
AOI-CM-004	The AOI Enabler SHALL be able to use the information of the access network type (e.g., Cellular network, WLAN), as well as the information of changes of access network (e.g. received as an event), used for the communication over the Shared AOI Connectivity.	1.0
AOI-CM-005	The AOI Enabler SHALL be able to use the information whether the device is roaming or not.	DELETE

5.5 AOI Client

Label	Description	Release
AOI-AC-001	The AOI Client SHALL support a Client Application wake-up mechanism, which can activate the application according to the notification from AOI Server, based on the policy defined in AOI Client.	1.0
AOI-AC-002	The AOI Enabler SHALL enable the AOI Client to get information (e.g. discovery, OTA provisioning) about the AOI Server.	Future Release
AOI-AC-003	The AOI Client SHALL support configuration of the AOI Server address.	1.0
AOI-AC-004	An AOI Client SHALL be able to provide the version of the AOI Enabler it uses when communicating with an AOI Server	Future Release
AOI-AC-005	The AOI Client SHALL support a mechanism to configure Shared AOI Connectivity	1.0
AOI-AC-006	AOI Enabler MAY support multiple AOI Clients on the same device	1.0
AOI-AC-007	If multiple AOI Clients are supported on the same device, then the AOI Enabler SHALL ensure AOI Client can be identified as active or inactive and support limitations on the number of active AOI Clients on that device at the same time.	Future Release
AOI-AC-008	The AOI enabler MUST support the configuration of relationships between AOI Server and AOI Client.	1.0
AOI-AC-009	The relationship between AOI Client and AOI Server SHALL be independent of AOI Enabled Application.	1.0
AOI-AC-010	The AOI Client SHALL provide APIs for Client Applications to use AOI services (such as registration, etc.).	1.0
AOI-AC-011	The AOI Enabler SHALL be able to update the AOI Client to a newer version of AOI Enabler. **Informational Note**: This requirement is targeting updating for new versions of AOI Enabler, and not update related to bug fix, UI change, etc.	Future Release

5.6 AOI Server

Label	Description	Release
AOI-AS-001	The AOI Enabler SHALL make the information about the appropriate AOI Server available for the Server Application (e.g. communication between AOI Server and Server Application).	1.0

AOI-AS-002	The AOI Server SHALL be able to provide the version of the AOI Enabler it uses when communicating with an AOI Client	Future Release

5.7 Security

Label	Description	Release
AOI-SEC-001	The AOI Enabler SHALL support mutual authentication between AOI Client and AOI Server.	1.0
AOI-SEC-002	The AOI Enabler SHALL support authorization of Server Application and Client Application to interact with the AOI Enabler	1.0
AOI-SEC-003	The AOI Enabler SHALL support a secure mechanism to generate and distribute AOI Subscription Information to prevent the identifier impersonation/spoofing.	Future Release
AOI-SEC-004	The AOI Enabler SHALL support secure exchange of data between Server Application and Client Application via AOI Server and AOI Client.	1.0
AOI-SEC-005	The AOI Enabler SHALL enable end-users to report malicious notifications to the AOI Server received from different applications [SpamReport]. **Information:** AOI 1.0 supports access to OMA SpamReport enabler required to report malicious notifications [SpamReport]	1.0
AOI-SEC-006	The AOI Enabler SHALL support the AOI Server to add/delete malicious applications to/from the blacklists [SpamReport].	1.0
AOI-SEC-007	The AOI Enabler SHALL be able to block notifications from blacklisted applications.	1.0
AOI-SEC-008	The AOI Enabler SHALL NOT expose any end-user identifiable data (e.g. Private IP address (es) being assigned to the device, IMSI, IMEI(SV), IMPU, IMPI and MSISDN) to the Client and Server Application.	1.0
AOI-SEC-009	The AOI Enabler SHOULD be able to support the authentication between Client Application and Server Application with the help of AOI Client and AOI Server, only if user permits and any user-private information is not revealed.	DELETE
AOI-SEC-010	The AOI Enabler SHOULD support different security mechanisms as applicable.	1.0

5.8 Overall System Requirements

Label	Description	Release
AOI-SYS-001	The AOI Enabler SHALL enable the AOI Server to indicate that it is getting overloaded.	1.0
AOI-SYS-002	The AOI Enabler SHALL provide mechanisms to prevent the AOI Server from being overloaded.	1.0
AOI-SYS-003	The AOI Enabler SHOULD support standard protocol and bearer binding whenever possible.	DELETE
AOI-SYS-004	The AOI Enabler SHALL enable both connection-less and connection-oriented mechanisms to deliver notifications.	1.0
AOI-SYS-005	The AOI Enabler SHOULD leverage existing technologies and specifications (e.g. protocols, data representations and encoding formats) defined within OMA or by other standards bodies.	1.0
AOI-SYS-006	The AOI Enabler SHALL be able to measure its performance when applicable (e.g. it can measure the notification frequency, delivery failure).	1.0
AOI-SYS-007	The AOI Enabler SHOULD make the data of its performance measurement available to AOI Client and AOI Server for later use if needed.	Future Release
AOI-SYS-008	The AOI Enabler SHALL be able to configure the criteria for performance measurement, e.g. the threshold of notification frequency.	Future Release
AOI-SYS-009	The AOI Enabler SHALL allow scalability	DELETE

5.9 Charging

Label	Description	Release
AOI-CHAR-001	The AOI Enabler SHOULD support charging mechanism.	1.0

6. Architectural Model

6.1 Dependencies

The AOI Enabler has no dependencies

6.2 Architectural Diagram

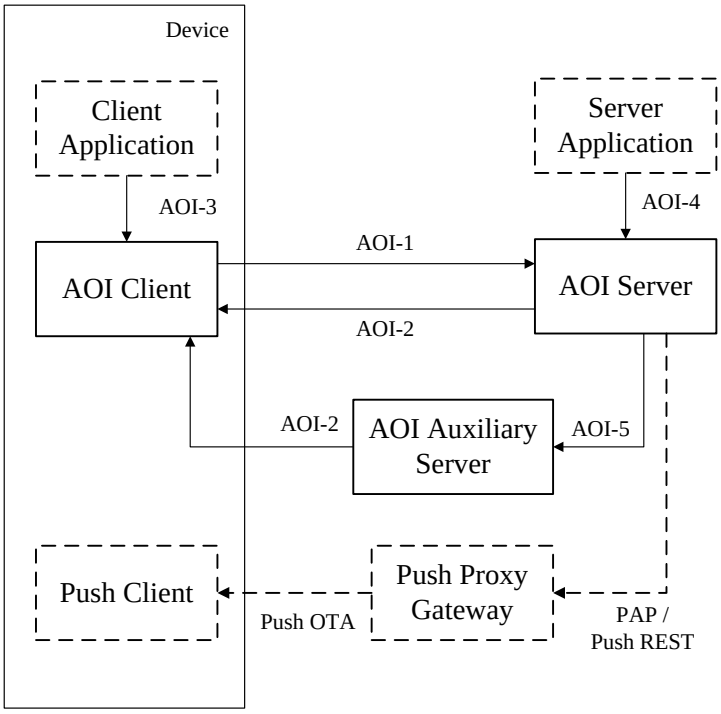

Figure 1: Overall Architecture Diagram

The above diagram depicts all the architecture elements of the AOI Enabler, as well as external entities and supporting enablers that may interact with the AOI Enabler.

Note that not all the elements and interfaces in the diagram above will apply to each notification channel or scenario, but they will apply based on the types of notifications channels for specific scenario. The architecture diagrams applied in specific scenarios are listed as following:

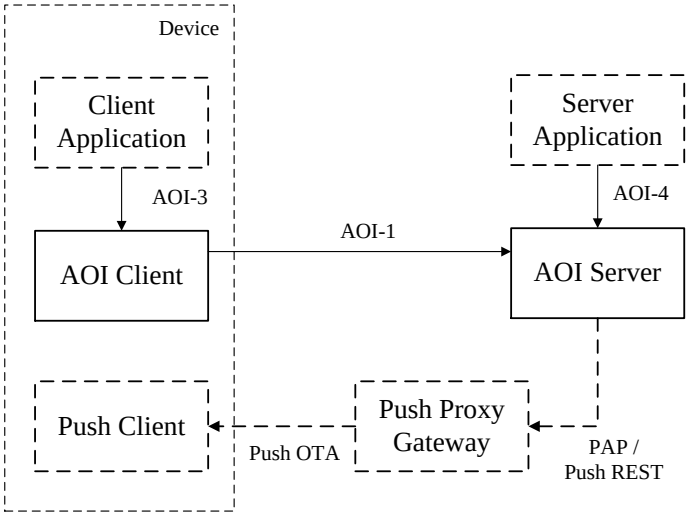

Figure 2: Architecture Diagram when using Push over SMS Mechanism

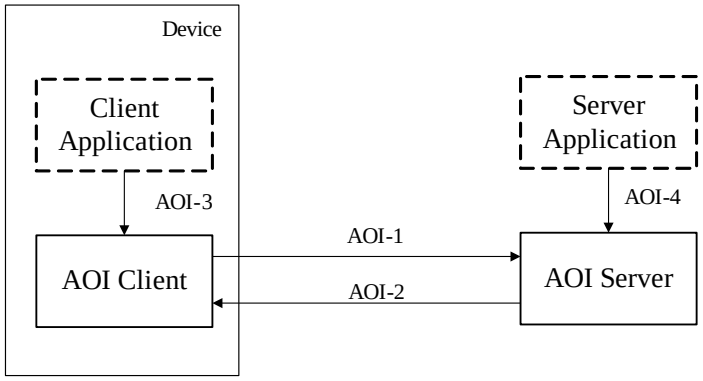

Figure 3: Architecture Diagram when not using AOI Auxiliary Server

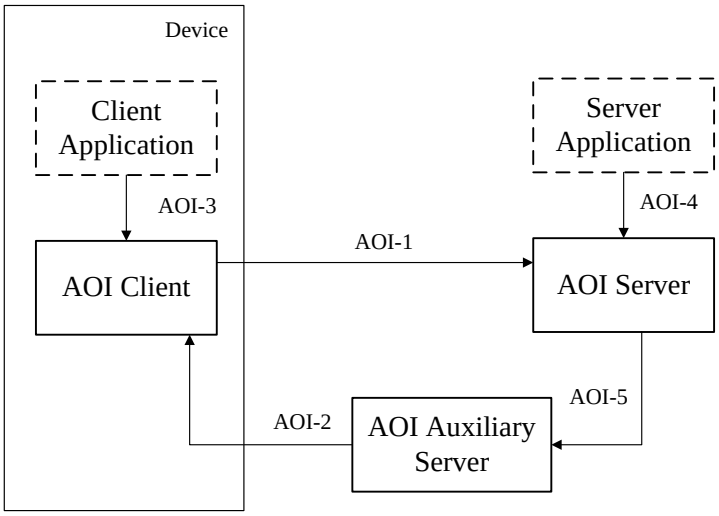

Figure 4: Architecture Diagram when using AOI Auxiliary Server

6.2.1 Notification Channel

6.2.1.1 Notification Channel based on Push over SMS

The Notification Channel based on Push over SMS can be used by the AOI Server to deliver notifications to the AOI Client by transferring them in the payload of the Push message. The architecture is depicted as Figure 2. This Notification Channel can be used regardless of the data connection used by the device (e.g. mobile connection, Wi-Fi), and whether the AOI Server is placed within or outside the MNO's network.

The interfaces with components of the Push Enabler are depicted in dotted line due to the fact that they are defined in the frame of the Push Enabler. However this specification defines some details as to how these interfaces are used, namely the Application ID associated to the AOI Enabler and the encoding of the notification to be transferred over the Push Enabler.

6.2.1.2 Client-Initiated TCP Notification Channel

The Client-initiated TCP Notification Channel can be used by the AOI Client to establish a connection with AOI Server and then receive notifications from AOI Server. The architecture is depicted as Figure 3. This Notification Channel is intended to be used when the device is within a Wi-Fi or another network with Firewall/NAT and Cellular data channels are disabled due to different policies.

The following diagram shows the architecture applicable to this case. Communication between AOI Client that is within the Wi-Fi or another network with Firewall/NAT and AOI Server can be realised over Wireless or Wired network and therefore Auxiliary Server is not applicable in this case.

6.2.1.3 Server-initiated TCP Notification Channel

The server-initiated TCP Notification Channel can be used by the AOI Server to establish a connection to the AOI Client and then deliver pending notifications to the AOI Client. This Notification Channel is intended to be used when the device uses the mobile data connection regardless whether the AOI Server is placed within or outside the MNO's network, but in the latter case a supporting AOI Auxiliary Server located within the MNO network is needed.

The Figure 4 depicts the architecture applicable to the case when AOI Server is located outside the MNO's network and an AOI Auxiliary Server is needed.

The Figure 3 depicts the architecture applicable to the case when AOI Server is within the mobile network.

6.2.1.4 Notification Channel based on UDP

The Notification Channel based on UDP can be used by the AOI Server to deliver notifications to the AOI Client by transferring them in the payload of the UDP messages. This Notification Channel can be used regardless of the data connection used by the device (e.g. mobile data connection or Wi-Fi), and whether the AOI Server is placed within or outside the MNO's network. The Figure 3 depicts the architecture applicable to the former case, and the Figure 4 depicts the architecture applicable to the latter case which needs a supporting AOI Auxiliary Server located within the MNO network.

6.2.2 Wake-Up mechanism

Wake-up mechanism will apply only if there is no notification channel established.

Wake-up mechanisms include UDP-based Wake-up and SMS-based Wake-up. It is up to AOI Client to decide based on different policies whether to initiate the TCP Notification Channel or not.

6.2.2.1 Notification Channel based on UDP Wake-up

The Notification Channel based on UDP Wake-up can be used by the AOI Server to wake-up the AOI Client and instruct it to establish a connection-oriented notification channel (i.e. TCP-based) with the AOI Server in order to receive pending notifications. This Notification Channel is intended to be used when the device uses the mobile data connection regardless whether the AOI Server is placed within or outside the MNO's network, but in the latter case a supporting AOI Auxiliary Server located within the MNO network is needed. The Figure 3 depicts the architecture applicable to the former case, and

the Figure 4 depicts the architecture applicable to the latter case which needs a supporting AOI Auxiliary Server located within the MNO network.

6.2.2.2 Notification Channel Based on Push over SMS Wake-up

The Notification Channel based on Push over SMS Wake-up can be used by the AOI Server to wake-up the AOI Client and instruct it to establish a connection-oriented notification channel (i.e. TCP-based) with the AOI Server in order to receive pending notifications. In this scenario, the AOI Server can be placed within or outside the MNO's network. The architecture is depicted as Figure 2.

6.3 Functional Components and Interfaces/reference points definition

6.3.1 Functional Components

6.3.1.1 Internal Functional Components

6.3.1.1.1 AOI Client

The AOI Client can include the following functionalities:

- managing the access from/to the Client Applications on the device related to the AOI Enabler functionalities
- receiving registration/de-registration from Client Applications
- sending registration(notification subscription)/de-registration request to AOI Server
- initiating Notification Channel setup with AOI Server
- receiving notifications from AOI Server
- receiving notification subscription from the Client Applications
- delivering notifications to the Client Applications managing Shared AOI Connectivity with AOI Server
- managing (e.g., add, remove, update) information related to Client Application
- managing (e.g. query, update) different status information (e.g. registered, de-registered ,unregistered) related to Client Application
- waking-up Client Applications when needed
- reporting the statistics information of the AOI Client (e.g. connect/disconnect activities, connection history, time period, number of notifications, number of kB sent or received, upload/download speed)
- alerting disaster/emergency notifications to the user (in case no Client Application Token is indicated in the notification)

6.3.1.1.2 AOI Server

The AOI Server can include the following functionalities:

- Registration Management:
  - receiving registration/de-registration from Server Applications;
  - receiving registration/de-registration from AOI Client;
- Notification Management:

- receiving notifications from Server Applications over secured connection;
- receiving registration(notification subscription)/de-registration request from the AOI Client;
- delivering notifications to the AOI Client, respecting the policies;
- controlling traffic for notification messages based on service provider policy, such as a threshold per application, selected applications or all applications.

- Connectivity Management:
  - maintaining AOI Connectivity if at least one Client Application is registered with the AOI Client.
  - managing (e.g., add, remove, update) information related to Client Applications and Server Applications in case the latter are registered;
  - managing (e.g. query, update) different status information (e.g. registered, de-registered ,unregistered) related to Client Application;
  - handling authentication of AOI Client;
  - handling authorization of Server Applications.
  - reporting and managing the statistics information of the AOI Server and AOI Clients(e.g. connect/disconnect activities, connection history, time period, number of notifications, number of kB sent or received, upload/download speed)
  - handling authorized Server Applications responsible for emergency situation,.
  - receiving lightweight messages from the AOI Client and delivering them to Server Applications.

6.3.1.1.3 AOI Auxiliary Server

The AOI Auxiliary Server is located within the MNO's network and provides support to the AOI Server, when this latter component lies outside of the MNO's network, in all those features that require a direct contact with the AOI Client from within the MNO's network. This element may be also used as a deployment option even if the AOI Server is placed within the MNO's network.

6.3.1.2 External Functional Components

6.3.1.2.1 Client Application

The Client Application can register to the AOI Client and receive notifications from the AOI Client.

Examples of Client Application can be web applications or native applications residing on the device.

6.3.1.2.2 Server Application

The Server Application can register to the AOI Server and send notifications to the AOI Server.

Examples of Server Application can be server side application residing on a network entity.

6.3.1.3 Supporting Enablers

6.3.1.3.1 Push Enabler

The Push Enabler supports the AOI Enabler in the delivery of notifications from the AOI Server to the AOI Client when the notification channel based on Push over SMS is used. The functionalities provided by the different components of the Push Enabler are described in [OMA-PUSH].

The AOI Server interacts with the Push Enabler via the Push Access Protocol (PAP) defined by the Push Enabler or via the RESTful Network API for OMA Push [OMA-PUSH-REST]

6.3.2 Interfaces

6.3.2.1 AOI-1

This interface is exposed by the AOI Server and can be used by the AOI Client to:

- register/de-register AOI Client with/from the AOI Server
- register/de-register Client Application with/from the AOI Server
- subscribe to notifications from the AOI Server
- report to the AOI Server about the alicious notification[SpamReport]
- reporting the result of notification delivery to the AOI Server
- initiate connection-oriented notification channel setup to AOI Server,(e.g. Client-initiated TCP Notification channel setup, triggered by UDP-based wake-up message etc.)
- report statistics information to the AOI Server, e.g. number of notifications received etc.
- deliver lightweight messages to the AOI Server
- activate, de-activate and maintain the notification channel with the AOI Server

6.3.2.2 AOI-2

This interface is exposed by the AOI Client and can be used by AOI Server and AOI Auxiliary Server to:

- deliver notifications to the AOI Client (not used by the AOI Auxiliary Server in case of Push over SMS wake-up notification channel);
- activate, de-activate and maintain the notification channel with the AOI Server

6.3.2.3 AOI-3

This interface is exposed by the AOI Client and can be used by the Client Applications to:

- register/de-register with/from the AOI Client;
- subscribe to and receive notifications from the AOI Client.
- query the AOI service status

6.3.2.4 AOI-4

This interface is exposed by the AOI Server and can be used by the Server Applications to:

- register/de-register with/from the AOI Server
- send notifications to the AOI Server
- request authorization for emergency notifications
- set an expiry time for the notifications to be delivered to the destination

6.3.2.5 AOI-5

This interface is exposed by the AOI Auxiliary Server and can be used by the AOI Server to:

- request AOI Auxiliary Server to send an indication to the AOI Client to initiate connection-oriented notification channel setup to AOI Server (in case of UDP wake-up notification channel)

- initiate connection-oriented notification channel setup (in case of server-initiated TCP notification channel) and notification channel based on UDP
- deliver notifications to the AOI Client (in case of server-initiated TCP notification channel)

7. Identification and Addressing

7.1 AOI Client ID

The AOI Client ID unambiguously identifies an AOI Client registered with an AOI Server. The AOI Client ID is generated during the AOI Client registration procedure and it is the responsibility of AOI Server to ensure its uniqueness in the realm of that specific AOI Server.

The AOI Client ID will be used by the AOI Server to identify a specific AOI Client and route notifications to the device in which that AOI Client is installed. The AOI Client ID should be unique for a specific device and valid from the moment it was allocated by a specific AOI Server until the AOI Client is deregistered from the AOI Server.

The information about AOI Client ID on AOI Server SHOULD include device identity.

7.2 Application ID

The Application ID unambiguously identifies an Application registered with an AOI Server. The Application ID is generated during the Server Application registration procedure. See the call flow G.11 for more details.

7.3 Client Application Token

The Client Application Token identifies the Client Application instance, or set thereof, that is intended to receive a type of notifications (e.g. Madrid weather updates, or new emails to bob@emailprovider.com), and is used by the Server Application to address the Client Application instance(s) target of each specific notification submitted to the AOI Server.

The Client Application Token SHALL be generated by the AOI Server upon receiving the Application registration request, according to the rules described in the Client Application Registration procedure.

7.4 Client Application Token Seed

The Client Application Token Seed is an optional parameter which the client Applications MAY convey at the Client Application registration request to indicate to the AOI Server to generate the Client Application Token based on the Client Application Token Seed. When present, the AOI Server SHALL generate the same Client Application Token as a response to Client Application registration requests coming from different instances of a same Application that contain a same Client Application Token Seed. The Client Application Token Seed is useful to indicate its intention to subscribe to a specific type of notifications (e.g. multicast notifications).

7.5 AOI Identification Relationship

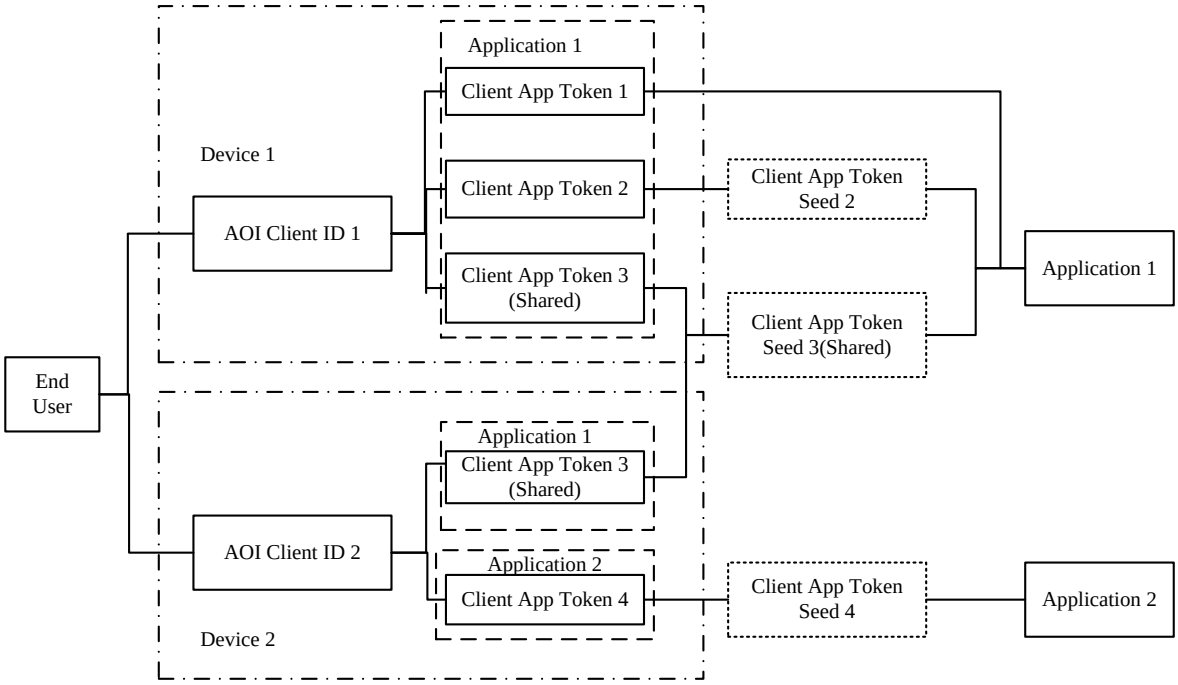

Figure 5: AOI ID relationship

The IDs and Tokens defined in AOI Enabler have following relationship between each other, as shown in Figure 5:

- An End User might have more than one device
- Each AOI Client on a device is associated with one AOI Client ID;
- Each AOI Client on a device might serve more than one Application, therefore is associated with more than one Client Application Token
- An Application might be associated with more than one Client Application Token
- An App Client Token might be shared among the multiple application instances that belong to the same Application installed on more than one device
- A Client Application Token must be associated with only one Client Application Token Seed if the Client Application Token is generated by Client Application Token Seed;

8. Functional Component Procedures

8.1 AOI Client Procedures

8.1.1 AOI Client Registration

The objective of the AOI Client registration is informing the AOI Server of its intention to use AOI services and requesting the AOI Client ID to be used in subsequent requests.

Upon installation/activation of the AOI Client or upon receiving a Client Application registration request whilst the AOI Client is not registered yet with the AOI Server, the AOI Client SHALL issue an AOI Client registration request to the AOI Server.

8.1.1.1 AOI Client Registration (Initial Registration)

The trigger of AOI Client Registration might depend on deployment policy and user configuration. Based on user policies, if SIM card binding to AOI Client is applied by the end user, the AOI Client Registration MAY include subscriber information (IMSI, MSISDN) from the SIM card

Upon receiving the AOI Client registration confirmation from the AOI Server, the AOI Client SHALL store the assigned AOI Client ID for use in future requests. If SIM card binding is applied during the registration, the AOI Client MAY store AOI Client ID on SIM card for future use.

Depending on user policies, if there is a binding between SIM card and AOI Client ID, AOI Client SHOULD be able to issue a new AOI Client registration request to obtain a new AOI Client ID from the AOI Server upon the change of SIM card..

8.1.1.2 AOI Client Registration (Re-registration)

The objective of the AOI Client re-registration is informing the AOI Server of its intention to continue with AOI services using the existing AOI Client ID.

If SIM card binding to AOI Client is applied by the end user, AOI Client MAY obtain AOI Client ID from SIM card if there is an associated AOI Client ID. The AOI Client Registration SHOULD include this AOI Client ID. The other procedure is same as initial AOI Client registration.

Upon receiving the AOI Client registration confirmation from the AOI Server, the AOI Client SHALL store the AOI Client ID confirmed in the response for use in future requests. If SIM card binding is applied during registration, the AOI Client MAY store AOI Client ID on SIM card for future use.

8.1.2 Notification Channel Activation

The objective of the Notification Channel Activation request is negotiating the notification channel(s) that may be used by the AOI Server to deliver notifications to the AOI Client.

The four types of notification channels described in section 6.2.1 may be proposed by the AOI Client:

The notification channels along with wake-up mechanisms described in section 6.2.2 allow the device where the AOI Client is running to remain in radio idle state as long as possible when it is connected through a mobile data connection, as they do not require sending keep-alive messages

Upon receiving a Client Application Registration request, when there is currently no notification channel active, the AOI Client MAY issue a Notification Channel Activation request, proposing at least one supported notification channel.

Upon receiving confirmation from the AOI Server of the selected notification channel(s), the AOI Client SHALL perform required actions to activate them, if needed (e.g. if OMA Push over SMS is used the AOI Client needs to register with the Push Client).

Upon any event that affects the notification channel(s) that can be used by the AOI Server to deliver notifications to the AOI Client (e.g. if the client gets a new IP address when the data connection switches between 3G and Wi-Fi or from one Wi-Fi to another), the AOI Client SHALL issue a Notification Channel Activation request to the AOI Server to indicate the bearer change of the notification channel(s).

8.1.3 Notification Channel Maintenance

The objective of the Notification Channel Maintenance request is maintaining the notification channel(s) that may be used by the AOI Server to deliver notifications to the AOI Client in those cases where the notification channel would be released after some inactivity period. An example of notification channel where the Notification Channel Maintenance is necessary is a connection-oriented notification channel that needs to traverse a Firewall/NAT.

When there is currently no notification messages for some time, e.g. 9 minutes, the AOI Client MAY issue a Notification Channel Maintenance request.

Upon receiving confirmation from the AOI Server to the Notification Channel Maintenance request, the AOI Client SHALL perform required actions to keep the notification channel active.

8.1.4 Notification Channel De-activation

The objective of the Notification Channel De-activation request is to deactivate the notification channel(s) between AOI Client and AOI Server.

The AOI Client MAY issue a Notification Channel De-activation request based on Service Provider policy or user defined policy.

Upon receiving confirmation of the selected notification channel(s) from the AOI Server, the AOI Client SHALL perform required actions to deactivate them, if needed.

8.1.5 Client Application Registration

The objective of the Client Application registration is to indicate the intent of Client Application to make use of the AOI Enabler capabilities and subscribe for reception of a specific type of notifications submitted by the Server Application and filtered by Client Application Token Seeds.

Upon receiving the Client Application registration requests from the Client Application, the AOI Client SHALL forward this request to the AOI Server, including the Client ID and Client Application Token Seed, if the Client Application issues one.

Upon receiving confirmation from the AOI Server of the successful registration the AOI Client SHALL store the result (mapping between Client Application Token and Client Application instance) and SHALL inform the Client Application about the result of the registration.

8.1.6 Notification Delivery

Upon receiving a notification request from the AOI Server the AOI Client SHALL forward the notification to the Client Application instance matching the Client Application Token provided in the notification request, if the Client Application is running or otherwise configured to be woken up.

8.1.6.1 Notification Compression

For the notification compression purpose, the AOI Client MAY report whether it supports compression/decompression and the supported compression algorithm(s) to the AOI Server. When receiving the compressed notification, the AOI Client will decompress the notification before sending it to the Client Application.

8.1.6.2 Notification and Wake-Up Aggregation

Depending on the AOI Client policy, when receiving non real-time notifications, the AOI Client MAY decide to wait for more notifications within a specific time interval and aggregate the delivery of the notifications to Client Applications in order to minimize the signaling load due to the service interaction triggered by device after receiving the notifications.

8.1.6.3 Notification Delivery Priority Handling

The AOI Client SHOULD support delivery of notifications according to the priority levels of the received notifications, by service classes. When receiving a notification, the AOI Client SHOULD check the priority level and deliver the notification accordingly. In the disaster conditions, the AOI Client SHOULD support higher prioritized delivery (e.g. a user interface notification or advancement in the queue if applicable) of notifications based on disaster/emergency service class. When receiving the disaster/emergency class of notification, regardless of the condition, the AOI Client will immediate delivery by the fastest and reliable way to the Client Application.

8.1.6.4 Notification Delivery Expiration

The objective of the Notification Delivery Expiration is to stop delivering the notifications after the expiry time. When receiving notifications, the AOI Client MAY decide to expire the notifications after the expiry time set by the Server Application.

8.1.7 AOI Client De-registration

The objective of the AOI Client de-registration is to release corresponding resources and inform the related parties that no service could be continued any more. There are two types of AOI client de-registration procedures:

- Initiated by AOI Client to inform the AOI Server of its intention to stop making use of AOI services
- Initiated by the AOI server to inform certain or all AOI Clients of Server's intention to stop providing the AOI service.

Before de-registration and/or other scenarios requiring de-activation of the AOI Client whilst the AOI Client is registered with the AOI Server, the AOI Client SHOULD issue an AOI Client de-registration request to the AOI Server, including AOI Client ID.

Upon receiving the AOI Client de-registration confirmation from the AOI Server, the AOI Client SHALL inform all the Client Application registered with the AOI Client about the result of the de-registration.

In case of procedure initiated by AOI Server, upon receiving the notification of de-registration, the AOI Client SHALL send acknowledgement message to AOI Server.

8.1.8 Client Application De-registration

The objective of the Client Application de-registration is to indicate the Client Application's intent to stop making use of the AOI Enabler capabilities.

Upon receiving the Client Application de-registration request from the Client Application, the AOI Client SHALL forward this request to the AOI Server, including Client Application Token and AOI Client ID.

Upon receiving confirmation from the AOI Server of the successful de-registration, the AOI Client SHALL delete the Client Application Token.

8.1.9 Authorized Emergency Notification

The objective of the emergency notification request is alerting the emergency notification(s) that may be used by the authorized Server Application to deliver disaster/emergence notifications to the user (in case no Client Application Token is indicated in the notification).

Upon receiving an emergency notification requests from the AOI Server, the AOI Client SHALL alert the notification immediately to the user. Emergency notification requests are used for broadcasting of messages to one or many groups of people "notifying" or alerting a group of individuals of a pending or existing emergency situation (e.g. tornadoes, tsunami, air-raid).

8.2 AOI Server Procedures

8.2.1 AOI Client Registration

Upon receiving the AOI Client registration requests from the AOI Client the AOI Server SHALL:

- check if any AOI Client ID is included in AOI Client Registration request, if NO:

  - check if any subscriber information (IMSI, MSISDN) included in the request, if NO:

    -handle the request as initial AOI Client Registration, assign a new AOI Client ID for this AOI Client Registration;

  - if YES:

    -check if simBinding in the request is true, if NO: handle the request as initial AOI Client Registration; assign a new AOI Client ID for this AOI Client Registration;

    -if YES, check internally in AOI Server if there is any AOI Client ID associated with given subscriber information, if no, handle the request as initial AOI Client Registration, assign a new AOI Client ID for this AOI Client Registration; store the subscriber information associated with the AOI Client ID;

      If there is an AOI Client ID associated with given subscriber information, return the AOI Client ID as result;

- if YES:

  - handle the request as re-registration, check if any subscriber information (IMSI, MSISDN) included in the request, if YES:

    -if simBinding is true and there is subscriber information associated with given AOI Client ID, and the subscriber information is different from that in the request, reject the request; notify AOI Client that it must initiate a new initial AOI Client Registration request.

    -if simBinding is false, handle the request as successful re-registration.

  If NO: handle the request as successful re-registration

AOI Server communicates the result of the registration to the AOI Client including the AOI Client ID to be used by the AOI Client in future requests

The result of the AOI Client Registration (i.e. AOI Client ID) SHALL be stored on AOI Server for future usage.

8.2.2 Notification Channel Activation

Upon receiving the Notification Channel Activation requests from the AOI Client the AOI Server SHALL select the most appropriate notification channel(s) taking into account the information received from the AOI Client, it SHALL activate said notification channel and SHALL communicate to the AOI Client the selected notification channel(s).

Note that more than one notification channel can be activated simultaneously. For instance, a connection-oriented channel and another based on OMA Push over SMS can be simultaneously active, so that in case the data connection fails unexpectedly the latter channel can still be used without requiring a new Notification Channel Activation.

The result of the Notification Channel Activation (i.e. notification channel info) SHALL be stored on AOI Server for future usage.

8.2.3 Notification Channel Maintenance

Upon receiving the Notification Channel Maintenance requests from the AOI Client the AOI Server SHALL keep the notification channel active and SHALL communicate to the AOI Client the confirmation.

Note that more than one notification channel can be kept active simultaneously. For instance, a connection-oriented channel and another based on OMA Push over SMS can be simultaneously active, so that in case the data connection fails unexpectedly the latter channel can still be used.

8.2.4 Notification Channel De-activation

Upon receiving the Notification Channel De-activation request from the AOI Client, the AOI Server SHALL de-activate said notification channel.

The AOI Server MAY also initiate the Notification Channel De-activation request according to Service Provider's policy.

The result of the Notification Channel De-activation (i.e. notification channel info) MAY be stored on AOI Server for future usage.

8.2.5 Client Application Registration

Upon receiving the Client Application registration requests from the AOI Client, the AOI Server SHALL generate a Client Application Token as per the following rules:

- In case the Application registration request does not include a Client Application Token Seed on the intention to subscribe to a specific type of notifications (e.g. multicast notification), then the AOI Server SHALL ensure that the generated Client Application Token is unique in the realm of that AOI Server, so that it represents just the single Application instance being registered.
- If on the other hand it does include such indication, the Client Application Token Seed, the AOI Server SHALL generate a Client Application Token specific to that type of notifications so that,
    - any other instance of that Application gets the same Client Application Token if it indicates the intention to subscribe to the same type of notifications. Preferably this happens even if the different instances of the same Application register in AOI Servers run by different service providers.
    - a malicious Application cannot subscribe to that type of notifications even if it intercepts the Client Application Token

The AOI Server SHALL store the mapping between the generated Client Application Token and the AOI Client ID provided in the request, and SHALL confirm the successful registration to the AOI Client.

8.2.6 Notification Delivery

Upon receiving the notification request from the Server Application, the AOI Server SHALL look up for the AOI Client ID, or set thereof, associated to：

-the Client Application Token provided in the notification request；

- Application ID provided in the notification request;

Afterwards it SHALL forward the notification to the AOI Client(s) associated to the AOI Client ID(s).

8.2.6.1 Notification Compression

When receiving notifications from the Server Applications, the AOI Server MAY decide whether or not to compress the notification based on the information reported by the AOI Client and its local policy, and act accordingly.

8.2.6.2 Notification Aggregation

Depending on the AOI Server policy, when receiving non real-time notifications from Server Application, the AOI Server MAY wait for more notifications to be sent to the same device within a specific time interval and aggregate the delivery of the notifications to reduce the signaling load due to the notification delivery and service interaction triggered by Device after receiving the notification.

8.2.6.3 Notification Delivery Policy

Upon receiving notifications from the Server Applications, the AOI Server SHALL check the channel status. When there is no channel available, the AOI Server SHALL reject the notification; or the AOI Server SHALL store the received notification and deliver it as soon as the channel becomes available, according to policies or notification delivery parameters.

8.2.6.4 Notification Delivery Handling

The AOI Server SHALL support delivery of notifications according to the priority levels of the notifications sent by the Server Application. The Server Applications will assign a priority value for the notification before sending it to the AOI Server. When receiving the notification, the AOI Server SHALL check the priority level and deliver the notification accordingly. 1. If the priority level is not set by the Server Application, a default value with '1' SHALL apply.

In the disaster conditions, the AOI Server SHOULD support higher prioritized delivery of notifications based on disaster/emergency service class. When receiving the disaster/emergency class of notification from the Server Applications, regardless of the condition, the AOI Server will immediate deliver by the fastest and reliable way to the AOI Client.

The AOI Server SHALL support prioritized delivery of notifications based on service classes which means the AOI Server will differentiate the notifications and deliver them accordingly.

8.2.6.5 Notification Delivery Expiration

The objective of the Notification Delivery Expiration is to stop delivering the notifications after the expiry time. When receiving notifications from the Server Application, the AOI Server SHALL decide to expire the notifications after the expiry time set by the Server Application.

8.2.6.6 Notification payload size limitation enforcement

The intention of this enforcement is to ensure that oversized notification messages between AOI Server and AOI Client are stopped.

Upon the reception of notification from Server Application, AOI Server MAY check the notification payload size to ensure that oversized payloads are not forwarded to the AOI client. This minimizes the over the air data traffic for messages not supported by the AOI Client.

Note: How to set the message size policy and what entity does it is not part of AOI specification. AOI Server is used to enforce this policy when required.

8.2.7 AOI Client De-registration

In case of procedure initiated by AOI Client, upon receiving the AOI Client de-registration requests from the AOI Client, the AOI Server SHALL delete registration information (i.e. mapping between AOI Client ID and all the Client Application Tokens) of the AOI Client.

The AOI Server SHALL confirm the successful de-registration to the AOI Client.

In case of procedure initiated by AOI Server, the AOI Server SHOULD inform the AOI Client the service would be stopped.

8.2.8 Client Application De-registration

Upon receiving the Client Application de-registration request from the AOI Client, the AOI Server SHALL delete registration information (i.e. the mapping between AOI Client ID and the deregistered Client Application Token), and SHALL confirm the successful de-registration to the AOI Client.

8.2.9 Server Application Registration

The objective of the Server Application registration is to inform the AOI Server of its intention to start making use of the service the AOI Server provides.

The registration message includes Application Name and Service Provider Name. Upon receiving the Server Application registration requests from the Server Application and upon the successful registration, AOI Server SHOULD store the information and generate an Application ID. AOI Server SHOULD communicate the result of the registration to the Server Application including the Application ID and AOIServerURL to be used by Server Application in future requests. Subsequently, Server Application SHOULD store the assigned Application ID and AOIServerURL for use in future requests.

8.2.10 Client Application Service Query

The objective of the Client Application Service Query is to obtain the AOI Service status.

Upon receiving the AOI Service Query request from the Client Application, the AOI Client SHALL return the updated status of AOI Service. In case the AOI Server URL changes, a new aoiServerUrl should be included in the response.

8.2.11 Server Application De-registration

The objective of the Server Application de-registration is to stop making use of the AOI service that the AOI Server provides.

The Server Application De-registration can be triggered either by AOI Server or Server Application.

If it is triggered by Server Application:

-before de-activation of the Server Application whilst the Server Application is registered to the AOI Server, the Server Application SHALL issue a Server Application de-registration request to the AOI Server, including the Identifier of the Server Application.

-upon receiving the Server Application de-registration request from the Server Application, the AOI Server SHOULD delete the registration information of the Server Application if the De-registration Request is PERMANENT. The AOI Server MAY inform the AOI Client that the Server Application is unavailable if the De-registration Request is set as TEMPORARY.

-the AOI Server SHALL confirm the successful de-registration to the Server Application.

If it is triggered by AOI Server:

- AOI Server SHALL reject the notification message with 418 error code or

- AOI Server sends Server Application De-registration request to Server Application.

8.2.12 Authorized Emergency Notification

Upon receiving the emergency notifications requests from the authorized Server Application, the AOI Server SHALL deliver by the fastest way to all the AOI Clients.

Before sending the emergency notifications request to the AOI Server, the Server Application SHALL issue a Server Application authorization request to the AOI Server, including the identifier of the Server Application.

9. Interfaces

9.1 Interface AOI-1

9.1.1 AOI Client Registration

The following table describes the elements of an AOI Client Registration request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientRegistrationReq', identifying this message as a AOI Client Registration request
clientId	String	0...1	This element contains the AOI Client ID associated with subscriber information which is stored in SIM card.
deviceID	String	1	This element contains the device ID of which initiate the AOI Client Registration request The device ID unambiguously identifies a device
simBinding	Boolean	0...1	This element contains the value indicating whether to bind AOI Client ID with SIM card. True: SIM binding is applied False: no SIM binding is applied(default)
IMSI	String	0…1	This element contains the IMSI; must be present if SIM binding is true. This element is included in the request if user preference for subscription information binding is on.
MSISDN	String	0…1	This element contains the IMSI; This element is included in the request if user preference for subscription information binding is on.
Compression	String	0…1	This element contains the supported compression algorithm(s). If this parameter is present, the compression is supported by the AOI Client.

Table 1: AOI Client Registration request

The following table describes the elements of an AOI Client Registration response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientRegistrationRes', identifying this message as a AOI Client Registration response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • 409: Conflicted SIM: SIM is conflicted with given subscriber information. • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
desc	String	1	This attribute contains a textual description that indicates the outcome of the request. The default descriptions can be found after each error code in the description of the 'code' element.
clientId	String	0…1	This element contains the AOI Client ID assigned by the AOI Server to the AOI Client as a successful result of the AOI Client registration request.

Table 2: AOI Client Registration response

9.1.2 AOI Client De-registration

The following table describes the elements of an AOI Client De-registration request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientDeregistrationReq', identifying this message as a AOI Client Deregistration request
clientId	String	1	This element contains the AOI Client ID assigned by the AOI Server during the AOI Client Registration procedure.

Table 3: AOI Client De-registration request

The following table describes the elements of an AOI Client De-registration success response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientDeregistrationRes', identifying this message as a AOI Client Deregistration response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 4: AOI Client De-registration response

9.1.3 Notification Channel Activation

The following table describes the elements of a Notification Channel Activation request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelActivationReq', identifying this message as a Notification Channel Activation request.
clientId	String	1	This element contains the AOI Client ID

Element	Type	Cardinality	Description
notificationChannel	Channel	1…N	Each occurrence of this element contains the data associated to a notification channel supported by the AOI Client

Table 5: Notification Channel Activation request

The following table describes the elements of the Channel type

Element	Type	Cardinality	Description
channelType	String	1	Type of the notification channel supported by the AOI Client. Can have the following values: • "pushOTA": OMA Push over SMS • "UDP": UDP based channel • TCP:TCP-based channel
data	PushOTAChannelData (if type="pushOTA") or UDPChannelData (if type="UDP") or TCPChannelData(if type="TCP")	0…1	Contains the data necessary to establish the corresponding notification channel This can be omitted only if PushOTAChannelData is empty.

Table 6: Channel type

The following table describes the element of the PushOTAChannelData type

Element	Type	Cardinality	Description
MSISDN	String	0…1	MSISDN registered in the device where the AOI Client is running and to which notifications can be sent This information may not be explicitly conveyed by the AOI Client but obtained by other means (e.g. network based mechanisms)

Table 7: PushOTAChannelData type

The following table describes the elements of the UDPChannelData type

Element	Type	Cardinality	Description
IP	String	1	Private IP address (within the MNO's network) in use by the device where the AOI Client is running and to which wake-up or notification UDP packet can be sent.

Element	Type	Cardinality	Description
Port	String	1	Port to which wake-up or notification UDP packet can be sent
MCC	String	0…1	Mobile Country Code associated to the mobile subscription in use by the device where the AOI Client is running This information may not be necessary in some scenarios (e.g. AOI Server providing service to a single network)
MNC	String	0…1	Mobile Network Code associated to the mobile subscription in use by the device where the AOI Client is running This information may not be necessary in some scenarios (e.g. AOI Server providing service to a single network)

Table 8: UDPChannelData type

The following table describes the elements of the TCPChannelData type

Element	Type	Cardinality	Description
IP	String	1	Private IP address (within the MNO's network) in use by the device where the AOI Client is running and to which TCP connection is established.
Port	String	1	Port to which TCP connection is established
MCC	String	0…1	Mobile Country Code associated to the mobile subscription in use by the device where the AOI Client is running This information may not be necessary in some scenarios (e.g. AOI Server providing service to a single network)
MNC	String	0…1	Mobile Network Code associated to the mobile subscription in use by the device where the AOI Client is running This information may not be necessary in some scenarios (e.g. AOI Server providing service to a single network)

Table 9: TCPChannelData type

The following table describes the elements of a Notification Channel Activation response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelActivationRes', identifying this message as a Notification Channel Activation response.
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK').

Element	Type	Cardinality	Description
			In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '401': 'Unauthorized': The provided AOI Client ID is not valid or not correctly registered to the AOI Server. • '403': 'Forbidden': the notification channel types proposed by the AOI Client are not supported at all by the AOI Server, or not supported for the corresponding user (e.g. the AOI Server does not support the establishment of a notification channel with users located in the Mobile Network indicated by the MCC/MNC parameters) • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
desc	String	1	This attribute contains a textual description that indicates the outcome of the request. The default descriptions can be found after each error code in the description of the 'code' element.
channelType	String	0..N	Type of the notification channel(s) activated by the AOI Server as the successful result: • "pushOTA": OMA Push over SMS • "UDP": UDP based channel • "TCP": TCP based channel

Table 10: Notification Channel Activation response

9.1.4 Notification Channel De-activation

The following table describes the elements of a Notification Channel Activation request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelDeactivationReq', identifying this message as a Notification Channel Deactivation request.
clientId	String	1	This element contains the AOI Client ID

Element	Type	Cardinality	Description
channelType	String	1	Type of the notification(s) channel to be deactivated by the AOI Client. Can have the following values: • "pushOTA": OMA Push over SMS • "UDP": UDP based channel • TCP:TCP based channel

Table 11: Notification Channel De-activation request

The following table describes the elements of a Notification Channel De-activation response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelDeactivationRes', identifying this message as a Notification Channel Deactivation response.
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 12: Notification Channel De-activation response

9.1.5 Notification Channel Maintenance

The following table describes the elements of a Notification Channel Maintenance request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelMaintenanceReq', identifying this message as a Notification Channel Maintenance request.
clientId	String	1	This element contains the AOI Client ID

Element	Type	Cardinality	Description
notificationChannel	Channel	1…N	Each occurrence of this element contains the data associated to a notification channel supported by the AOI Client Please refer to Section 9.1.3 for Channel type definition.
maintenanceTime	Integer	0…1	This element indicates the maintenance time, measured in seconds.

Table 13: Notification Channel Maintenance request

The following table describes the elements of a Notification Channel Maintenance response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelMaintenanceRes', identifying this message as a Notification Channel Maintenance response.
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
desc	String	1	This attribute contains a textual description that indicates the outcome of the request.
maintenanceTime	Integer	0…1	This element indicates the maintenance time adjusted by AOI Server, measured in seconds. If this parameter is present, the AOI Client SHALL perform required actions to keep the notification channel active based on this parameter. If maintenanceTime parameter in the request is not present, the AOI Server SHALL include this parameter.

Table 14: Notification Channel Maintenance response

9.1.6 Client Application Registration

The following table describes the elements of a Client Application Registration request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'clientAppRegistrationReq', identifying this message as a Client Application Registration request
clientId	String	1	This element contains the AOI Client ID
tokenSeed	String	0...1	This element contains the Client Application Token Seed. Same clientAppToken will be generated based on the same tokenSeed.
ApplicationID	String	1	This element contains the Application ID, used by the AOI Server for message routing

Table 15: Client Application Registration request

The following table describes the elements of a Client Application Registration success response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'clientAppRegistrationRes', identifying this message as a Client Application Registration response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '401': 'Unauthorized': The provided AOI Client ID is not valid or not correctly registered to the AOI Server. • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
desc	String	1	This attribute contains a textual description that indicates the outcome of the request. The default descriptions can be found after each error code in the description of the 'code' element

Element	Type	Cardinality	Description
clientAppToken	String	1	This element contains the Client Application Token as the successful result, to be used by the Server Application for the notification addressing
aoiServerUrl	URL	0..1	This element contains the URL of the AOI Server to which notification requests need to be targeted. If this information is known by other means it does not need to be conveyed in the Client Application Registration response

Table 16: Client Application Registration response

9.1.7 Client Application De-registration

The following table describes the elements of a Client Application De-registration request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'clientAppDeregistrationReq', identifying this message as a Client Application Deregistration request
clientId	String	1	This element contains the AOI Client ID
clientAppToken	String	1	This element contains the Client Application Token

Table 17: Client Application De-registration request

The following table describes the elements of a Client Application De-registration response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'clientAppDeregistrationRes', identifying this message as a Client Application Deregistration response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.

Element	Type	Cardinality	Description
Desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 18: Client Application De-registration response

9.1.8 AOI Client statistics reporting

The following table describes the elements of a AOI Client statistics reporting request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'StatisticsReportingReq', identifying this message as an AOI Client statistics reporting request
StatsReportMessageID	String	1	Unique identifier to identify the stats report sent by the AOI Client
clientID	String	1	Indicates the unique identifier of the AOI Client.
statsData	String	1..n	Gives the list of structures that contain stats information. These structures, for example can contain, but not limited information about connect/disconnect activities, connection history, time period, number of notifications sent, number of kB sent, upload/download speed etc.

Table 19: Client Application Statistics reporting request

The following table describes the elements of an AOI Client statistics reporting response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'StatisticsReportingRes', identifying this message as a Client Application Deregistration response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.

Element	Type	Cardinality	Description
desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 20: AOI Client statistics reporting response

9.1.9 Lightweight Message Delivery

The following table describes the elements of a Lightweight Message Delivery request sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'lightweightMessageDeliveryReq', identifying this message as a Lightweight Message Delivery request
clientId	String	1	This element contains the AOI Client ID assigned by the AOI Server to the AOI Client as a result of the AOI Client Registration procedure.
ApplicationID	String	1	This element contains the Application Identifier, used by the AOI Server for message routing
payload	String	1	This element contains the payload of the lightweight message sent by the Client Application targeted to the Server Application.

Table 21: Lightweight Message Delivery request

The following table describes the elements of a Lightweight Message Delivery response sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'lightweightMessageDeliveryRes', identifying this message as a Lightweight Message Delivery response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '202' ('Accepted') . In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request. • '401': 'Unauthorized': The provided Application ID is not valid or not registered to the AOI Server. • '403': 'Forbidden': The requesting Application Client does not have the rights to send lightweight

Element	Type	Cardinality	Description
			messages. • '415': 'Unsupported Media Type': The payload type is not supported by the AOI Server. • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The AOI Client may re-attempt the operation.
Desc	String	1	This attribute contains a textual description that indicates the outcome of the request. The default descriptions can be found after each error code in the description of the 'code' element.

Table 22: Lightweight Message Delivery response

The Request and Response Definition for Lightweight Message Delivery will also be supported in the interface AOI-3 and AOI-4.

9.2 Interface AOI-2

9.2.1 Notification Channel De-activation

The following table describes the elements of a Notification Channel De-activation request sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelDeactivationReq', identifying this message as a Notification Channel Deactivation request.
ServerID	String	1	This element contains the AOI Server ID
channelType	String	1	Type of the notification(s) channel to be deactivated by the AOI Server. Can have the following values: • "pushOTA": OMA Push over SMS • "UDP": UDP based channel • TCP:TCP based channel

Table 23: Notification Channel De-activation request

The following table describes the elements of a Notification Channel De-activation response sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'channelDeactivationRes', identifying this message as a Notification Channel Deactivation response.

Element	Type	Cardinality	Description
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request
desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 24: Notification Channel De-activation response

9.2.2 Notification Delivery

The following table describes the elements of a Notification Delivery request sent from the AOI Server to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'notificationDeliveryReq', identifying this message as a Notification Delivery request
Notification	NotificationType	1…N	Each Notification element contains a notification associated to a single Client Application Registration; 1 or many Notification elements can be included in a single Notification Delivery request sent from the AOI Server to the AOI Client

Table 25: Notification Delivery request

The following table describes the elements of the NotificationType type:

Element	Type	Cardinality	Description
clientAppToken	String	1	This element identifies each specific client application registration and is to be used by the AOI Client to route the notification to the right Client Application.
payload		0…1	This element contains the payload of the notification sent by the Server Application targeted to the Client Application. If the payload element is absent, it means that there is some information available at the Server Application.
Priority	Enumeration	1	This element specifies the delivery priority of the notification message. The methods used to reduce delivery latency are implementation dependent, but may include the use of a different bearer or the reordering of messages waiting transmission to send the higher priority messages first

Element	Type	Cardinality	Description
IsRealtime	Boolean	0…1	This element contains the value indicating whether this notification is delivered in real time. True: real time(default) False: non real time
expiryTime	DateTimeStamp	0…1	Date and Time by which the notification will expire. The notification will not be expired by default value.

Table 26: NotificationType type

The following table describes the elements of a Notification Delivery response sent from the AOI Client to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'notificationDeliveryRes', identifying this message as a Notification Delivery response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '202' ('Accepted') In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '401': 'Unauthorized': The provided Client Application Token is not valid or not registered to the AOI Client • '415': 'Unsupported Media Type': The payload type is not supported by the AOI Client
desc	String	1	This attribute contains a textual description that indicates the outcome of the request. The default descriptors can be found after each error code in the description of the 'code' element.

Table 27: Notification Delivery response

9.3 Interface AOI-3

9.3.1 Client Application Registration

The following table describes the elements of a Client Application Registration request sent to the AOI Client:

Element	Type	Cardinality	Description

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'clientAppRegistrationReq', identifying this message as a Client Application Registration request
ApplicationID	String	1	This element contains the Application ID
tokenSeed	String	0...1	This element contains the Client Application Token Seed. Same clientAppToken will be generated given the same tokenSeed.

Table 28: ClientApplicationRegistration request

The following table describes the elements of a Client Application Registration response sent from the AOI Client to the Client Application:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'clientAppRegistrationRes', identifying this message as a Client Application Registration response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '401': 'Unauthorized': The provided AOI Client Application ID is not valid or not correctly registered to the AOI Server. • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
desc	String	1	This attribute contains a textual description that indicates the outcome of the request. The default descriptions can be found after each error code in the description of the 'code' element.
clientAppToken	String	0…1	This element contains the Client Application Token as the successful result, to be used by the Server Application for the notification addressing
aoiServerUrl	URL	0..1	This element contains the URL of the AOI Server to which notification requests need to be targeted. If this information is known by other means it does not need to be conveyed in the Client Application Registration response

Table 29: ClientApplicationRegistration response

9.3.2 AOI Client Application De-registration

The following table describes the elements of an AOI Client Application De-registration request sent from the AOI Client Application to the AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientAppDeregistrationReq', identifying this message as a AOI Client Application Deregistration request
clientAppToken	String	1	This element contains the Client Application Token, to be used by the Server Application for the notification addressing

Table 30: AOI Client Application De-registration request

The following table describes the elements of a De-registration response sent from the AOI Client to the AOI Client Application:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientAppDeregistrationRes, identifying this message as a AOI Client Application Deregistration response.
Code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
Desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 31: AOI Client Application De-registration response

9.3.3 AOI Service Query

The following table describes the elements of an AOI Service Query request sent from the Client Application to AOI Client:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'serviceQueryReq', identifying this message as a Service Query request
clientAppToken	String	1	This element contains the Client Application Token

Table 32: AOI Service Query

The following table describes the elements of an AOI Service Query success response sent from the AOI Client to the Client Application:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'serviceQueryRes', identifying this message as a Service Query response
Code	String	1	This attribute contains a code that indicates the outcome of the request. The Client Application SHALL use the previous AOI Service Information In case of a success response it SHALL be set to '200' In case of a failed response it SHALL be set to '404'
Desc	String	1	This attribute contains a textual description that indicates the outcome of the request.
aoiServerUrl	URL	0..1	This element contains the URL of the AOI Server to which notification requests need to be targeted. If this information is known by other means it does not need to be conveyed in the response

Table 33: AOI Service Query response

9.4 Interface AOI-4

The following table describes the elements of a Notification Delivery request sent from the Server Application to the AOI Server:

9.4.1 Notification Delivery

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'notificationDeliveryReq', identifying this message as a Notification Delivery request
clientAppToken	String	0…1	This element contains the Client Application Token, used by the Server Application for the notification addressing and to be used by the AOI Server for notification routing If not present, the notification is sent to all Client Application associated with given Application ID.
Payload	String	0…1	This element contains the payload of the notification sent by the Server Application targeted to the Client Application. If the payload element is absent, it means that there is some information available at the Server Application.
Priority	Enumeration	1	This element specifies the delivery priority of the notification message. The methods used to reduce delivery latency are implementation dependent, but may include the use of a different bearer or the reordering of messages waiting transmission to send the higher priority messages first

Element	Type	Cardinality	Description
ApplicationID	String	1	This element identifies the application which sent the notification.
IsRealtime	Boolean	0...1	This element contains the value indicating whether this notification is delivered in real time. True: real time(default) False: non real time
expiryTime	DateTimeStamp	0…1	Date and Time by which the notification will expire. The notification will not be expired by default value.

Table 34: Notification Delivery request

The following table describes the elements of a Notification Delivery success response sent from the AOI Server to the Server Application:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'notificationDeliveryRes', identifying this message as a Notification Delivery response
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '202' ('Accepted') In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request. • '401': 'Unauthorized': The provided Client Application Token is not valid or not registered to the AOI Server. • '403': 'Forbidden': The requesting Application Server does not have the rights to send notification (e.g. no right to use the requested priority). • '413': and the message size is above the limit: This error code would be applicable if AOI client has opted-in for message size control This error code would not be applicable if AOI client has opted-out for message size control applicable. • '415': 'Unsupported Media Type': The payload type is not supported by the AOI Client.

Element	Type	Cardinality	Description
			• '418': Server Application DeRegistered • 421:Trial Notification quota exceeded or trial notification delivery expired. • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.
Desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 35: Notification Delivery response

9.4.2 Server Application Registration

The following table describes the elements of a Server Application registration request sent from the Server Application to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'ServerAppRegistrationReq', identifying this message as a Server Application Registration request.
ApplicationName	String	1	This element contains the Application Name, which is only used until the Application ID is obtained.

Table 36: Server Application Registration request

The following table describes the elements of a Registration response sent from the AOI Server to the Server Application:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'ServerAppRegistrationRes', identifying this message as a Server Application Registration response
code	String	1	This attribute contains a code that indicates the outcome of both requests, to register and to send emergency notifications. In case of a success for both requests, the response SHALL be set to '200' ('OK') In case of a success for registration request, but not the authorization request, the response SHALL be set to '201' . In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Server is temporarily not ready to process the request. The client may re-attempt the operation.

Element	Type	Cardinality	Description
desc	String	1	This attribute contains a textual description that indicates the outcome of the request , including the outcome for authorization request for emergency notifications. The default descriptions can be found after each error code in the description of the 'code' element.
ServerApplicationId	String	1	This element contains the Application ID assigned by the AOI Server to the Server Application as the successful result.
AOIServerURL	URL	0..1	This element contains the URL of the AOI Server to which notification requests need to be targeted. If this information is known by other means(i.e. it is passed to the Client Application in the "Client Application registration" process) it does not need to be conveyed in the Server Application Registration response

Table 37: Server Application Registration response

9.4.3 Server Application De-registration

9.4.3.1 Server Application De-registration (Server Application triggered)

The following table describes the elements of a Server Application de-registration request sent from the Server Application to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'serverAppDeregistrationReq, identifying this message as a Server Application Deregistration request.
ApplicationID	String	1	This element contains Application ID assigned by the AOI Server to the Server Application.
Desc	String	1	Two types of deregistration exists: TEMPORARY –when the Server Application is expected to re-register within the next few hours, e.g. going off-line due to maintenance purpose. PERMANENT – when the Server Application is being deactivated due to (a) termination of the service, (b) switch to a different AOI Server, or (c) switch to polling method for informing new updates.

Table 38: Server Application De-registration request

The following table describes the elements of a De-registration response sent from the AOI Server to the Server Application:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'serverAppDeregistrationRes, identifying this message as a Server Application Deregistration response.
code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK')

Table 39: Server Application De-registration response

9.4.3.2 Server Application De-registration (AOI Server triggered)

The following table describes the elements of a AOI Server triggered Server Application de-registration request sent from the AOI Server to Server Application:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'DeregisterServerAppReq', identifying this message as a Server Application Deregistration request.
ApplicationID	String	0...1	This element contains Application ID assigned by the AOI Server to the Server Application
desc	String	1	Two types of deregistration exists: TEMPORARY –when the Server Application is expected to re-register within the next few hours, e.g. going off-line due to maintenance purpose. PERMANENT – when the Server Application is being deactivated due to (a) termination of the service, (b) switch to a different AOI Server, or (c) switch to polling method for informing new updates.
Reason	String	1	This element contains the reason for the deregistration

Table 40: Server Application De-registration request (AOI Server triggered)

The following table describes the elements of a De-registration response, which is optional, sent from the Server Application to AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'serverAppDeregistrationRes', identifying this message as a Server Application Deregistration response.
Code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200' ('OK')

Table 41: Server Application De-registration response (AOI Server triggered)

9.5 Interface AOI-5

9.5.1 Notification Delivery

The following table describes the elements of a Notification Delivery request sent from the AOI Server to the AOI Auxiliary Server. This operation over AOI-5 interface only applies to the TCP-based Notification Channel case:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'notificationDeliveryReq', identifying this message as a Notification Delivery request
Notification	NotificationType	1…N	Each Notification element contains a notification associated to a single Client Application Registration; 1 or many Notification elements targeted to a single AOI Client can be included in a single Notification Delivery request sent from the AOI Server to the AOI Auxiliary Server.
IP	String	1	Private IP address (within the MNO's network) of the device in which the AOI Client target of the notification is running. The TCP-based Notification Channel is established with the IP address indicated in this element and the port indicated in the 'port' element
Port	String	1	Port of the device in which the AOI Client target of the notification is running and that is used to establish the TCP-based Notification Channel. The TCP-based Notification Channel is established with the IP address indicated in the 'IP' element and the port indicated in this element.

Table 42: Notification Delivery request

The following table describes the elements of a Notification Delivery response sent from the AOI Auxiliary Server to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'notificationDeliveryRes', identifying this message as a Notification Delivery response
Code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '202' ('Accepted') . In case of an error response one of the following codes MAY be used:

Element	Type	Cardinality	Description
			• '400': 'Bad Request': Bad syntax in the request. • '403': 'Forbidden': The requesting Application Server does not have the rights to send notification (e.g. no right to use the requested priority). • '415': 'Unsupported Media Type': The payload type is not supported by the AOI Client. • '503': 'Service Unavailable': The AOI Auxiliary Server is temporarily not ready to process the request. The client may re-attempt the operation.
Desc	String	1	This attribute contains a textual description that indicates the outcome of the request. The default descriptors can be found after each error code in the description of the 'code' element.

Table 43: Notification Delivery response

9.5.2 AOI Client Wake-up

The following table describes the elements of an AOI Client Wake-up request sent from the AOI Server to the AOI Auxiliary Server. This operation over AOI-5 interface only applies to the Notification Channel based on UDP Wake-up case:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientWakeupReq', identifying this message as a AOI Client Wake-up request.
IP	String	1	Private IP address (within the MNO's network) in use by the device where the AOI Client is running and to which wake-up UDP packet can be sent.
port	String	1	Port of the device where the AOI Client is running and to which wake-up UDP packet can be sent.

Table 44: AOI Client Wake-up request

The following table describes the elements of an AOI Client Wake-up response sent from the AOI Auxiliary Server to the AOI Server:

Element	Type	Cardinality	Description
messageType	String	1	Element with value set to 'aoiClientWakeupRes', identifying this message as a AOI Client Wake-up response
Code	String	1	This attribute contains a code that indicates the outcome of the request. In case of a success response it SHALL be set to '200'

Element	Type	Cardinality	Description
			('OK'). In case of an error response one of the following codes MAY be used: • '400': 'Bad Request': Bad syntax in the request • '503': 'Service Unavailable': The AOI Auxiliary Server is temporarily not ready to process the request. The AOI Server may re-attempt the operation.
Desc	String	1	This attribute contains a textual description that indicates the outcome of the request.

Table 45: AOI Client Wake-up response

9.6 Common Data Structures Enumerations

9.6.1 Priority

Note: the enumerations of Priority, including but not limited to:

Enumeration	Description
0	Low : The value "low" indicates the slowest delivery
1	Medium (Default)
2	High : The value "high" indicates a faster delivery is desired
3	Urgent – in case of disaster/emergency. It indicates the fastest delivery is desired

Table 46: Priority Enumeration

10.Protocol Binding (Informative)

Some of the AOI interfaces (e.g.AOI-1, AOI-2, AOI-5) can be implemented via different protocol bindings (e.g. UDP, TCP, WebSockets), and the interchange message/data can be described by specific format (e.g. JSON, XML). The following sub-section will use JSON for example.

10.1 AOI-1 Interface

10.1.1 AOI Client Registration

Whenever an AOI Client wants to perform the AOI Client Registration it firstly establishes a websocket with the AOI Server and then issues an AOI Client Registration request, encoded as a JSON-RPC 2.0 [JSON-RPC] request object, with:

- “method” member set to “aoiClientRegistrationReq”
- “id” member including an unique identifier for each request
- “params” member including the following parameter:
    - “deviceNumber” : mandatory, string

Example:

```
{
    "jsonrpc": "2.0",
    "method": "aoiClientRegistrationReq",
    "id": "111",
    "params": {
        deviceNumber: “54654654168468654 65”
        }
}
```

Upon successful registration the AOI Server returns an AOI Client Registration success response encoded as a JSON-RPC 2.0 [JSON-RPC] response object, with:

- “id” member matching the identifier included in the request object
- “result” member including the following parameters:
    - “clientID”: mandatory, string
    - “code”: mandatory, string
    - “message”: mandatory, string. Corresponds to the “desc” parameter

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "111",
    "result": {
        clientID: “aoi123456789“,
        code: 200,
        message: “OK“
```

```
        }
}
```

Then the AOI Client keeps the websocket open so that a Notification Channel Activation request can be sent.

Upon unsuccessful registration the AOI Server returns an AOI Client Registration error response encoded as a JSON-RPC 2.0 [JSON-RPC] response object, with:

- "id" member matching the identifier included in the request object
- "error" member including the following parameters:
  - "code": mandatory, string
  - "message": mandatory, string. Corresponds to the "desc" parameter

Example of error response:

```
{
    "jsonrpc": "2.0",
    "id": "111",
    "error": {
        code: 503,
        message: "Service Unavailable"
        }
}
```

Then the AOI Client may either retry the request if the error may be temporary or otherwise close the websocket.

10.1.2 AOI Client De-registration

Whenever an AOI Client wants to perform the AOI Client De-registration it firstly checks if there is an open websocket with the AOI Server and if not then it establishes it. Then it issues an AOI Client De-registration request, encoded as a JSON-RPC 2.0 [JSON-RPC] request object, with:

- "method" member set to "aoiClientDeregistrationReq"
- "id" member including an unique identifier for each request
- "params" member including the following parameter:
  - "clientID" : mandatory, string

Example:

```
{
    "jsonrpc": "2.0",
    "method": "aoiClientDeregistrationReq",
    "id": "112",
    "params": {
        clientID: "aoi123456789"
        }
```

```
}
```

Then the AOI Server returns an AOI Client De-registration response encoded as a JSON-RPC 2.0 [JSON-RPC] response object, with:

- "id" member matching the identifier included in the request object
- "result" or "error" member, in case of success or error response respectively, including the following parameters:
  - "code": mandatory, string
  - "message": mandatory, string. Corresponds to the "desc" parameter

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "121",
    "result": {
        code: 200,
        message: "OK"
        }
}
```

Example of error response:

```
{   "jsonrpc": "2.0",
    "id": "121",
    "error": {
        code: 503,
        message: "Service Unavailable"
        }
}
```

Then the AOI Client proceeds to close the websocket.

10.1.3 Notification Channel Activation

Whenever an AOI Client wants to perform the Notification Channel Activation it firstly checks if there is an open websocket with the AOI Server and if not then it establishes it. Then it issues an Notification Channel Activation request, encoded as a JSON-RPC 2.0 [JSON-RPC] request object, with:

- "method" member set to "channelActivationReq"
- "id" member including an unique identifier for each request
- "params" member including the following parameter:
  - "clientID" : mandatory, string
  - "notificationChannel" member including the following parameter:
    - "channelType": mandatory, string
    - "data" member:

if "channelType" equals "pushOTA", "data" member including the following parameter:

- "MSISDN": string

if "channelType" equals "UDP" or "TCP", "data" member including the following parameter:

- "IP": mandatory, string
- "port": mandatory, string
- "MCC": string
- "MNC": string

Example:

```
{
    "jsonrpc": "2.0",
    "method": "channelActivationReq",
    "id": "113",
    "params": {
       "clientID": "aoi123456789",
       "notificationChannel":[{
                        "channelType": "pushOTA" ,
                        "data":{
                                "MSISDN": "8618611111111"
                    }
                        },{
                        "channelType": "UDP" ,
                        "data":{
                                "IP": "10.10.10.10",
                                "port": "1111",
                                "MCC": "460",
                                "MNC": "01"
                    }
                }
                                                                    ]
        }
}
```

Then the AOI Server returns an Notification Channel Activation response encoded as a JSON-RPC 2.0 [JSON-RPC] response object, with:

- "id" member matching the identifier included in the request object
- "result" or "error" member, in case of success or error response respectively, including the following parameters:
    - "code": mandatory, string
    - "message": mandatory, string. Corresponds to the "desc" parameter
    - "channelType": array of string. In case of error response, the parameter is absent

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "113",
    "result": {
        “code”: 200,
        “message”: “OK“,
        “channelType”: [
                                “pushOTA” ,
                                “UDP”
                ]
        }
}
```

Example of error response:

```
{   "jsonrpc": "2.0",
    "id": "113",
    "error": {
        code: 503,
        message: “Service Unavailable“
        }
}
```

10.1.4 Notification Channel De-activation

When the AOI Client wants to deactivate the notification channel with the AOI Server, it will send a Notification Channel Deactivation request, which could be encoded as, e.g. JSON-RPC 2.0 [JSON-RPC] request object, with:

- “method” member set to “ChannelDeactivationReq”
- “id” member including an unique identifier for each request
- “params” member including the following parameters:
    - “clientID” : mandatory, string
    - “ChannelType”: mandatory, string, =[“pushOTA”, “UDP”, “TCP”]

Example:

```
{
                                "jsonrpc": "2.0",
                                "method": "ChannelDeactivationReq",
                                "id": "114",
                                "params": {
```

```
                clientID: “aoi123456789”,
                ChannelType: “TCP”
        }
}
```

The AOI Server returns a Notification Channel Deactivation response, which could be encoded as, e.g. JSON-RPC 2.0 [JSON-RPC] response object, with:

- “id” member matching the identifier included in the request object
- “result” or “error” member, in case of success or error response respectively, including the following parameters:
    - “code”: mandatory, string
    - “message”: mandatory, string. Corresponds to the “desc” parameter

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "114",
    "result": {
        code: 200,
        message: “OK“
        }
}
```

Example of error response:

```
{   "jsonrpc": "2.0",
    "id": "114",
    "error": {
        code:400,
        message: “Bad Request“
        }
}
```

10.1.5 Client Application Registration

Whenever an AOI Client wants to perform the Client Application Registration it firstly checks if there is an open websocket with the AOI Server and if not then it establishes it. Then it issues an Client Application Registration request, encoded as a JSON-RPC 2.0 [JSON-RPC] request object, with:

- "method" member set to "clientAppRegistrationReq"
- "id" member including an unique identifier for each request
- "params" member including the following parameter:
    - "clientID" : mandatory, string
    - "ApplicationID": mandatory, string
    - "tokenSeed" : optional, string

Example:

```
{
    "jsonrpc": "2.0",
    "method": "clientAppRegistrationReq ",
    "id": "116",
    "params": {
        "clientID": "aoi123456789",
        "ApplicationID": "app123456789"
        "tokenSeed": "tokenseed123456789";
      }
}
```

Then the AOI Server returns an Client Application Registration response encoded as a JSON-RPC 2.0 [JSON-RPC] response object, with:

- "id" member matching the identifier included in the request object
- "result" or "error" member, in case of success or error response respectively, including the following parameters:
    - "code": mandatory, string
    - "message": mandatory, string. Corresponds to the "desc" parameter
    - "clientAppToken": mandatory, string
    - "aoiServerUrl": optional, URL

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "116",
    "result": {
        "code": 200,
        "message": "OK",
        "ClientAppToken": "token123456789";
        "aoiServerUrl": "http://10.10.10.10:80/aoiServer"
        }
```

```
}
```

Example of error response:

```
{   "jsonrpc": "2.0",
    "id": "116",
    "error": {
          code: 503,
          message: “Service Unavailable“
          }
}
```

10.1.6 Client Application De-registration

Whenever an AOI Client wants to perform the Client Application De-registration it firstly checks if there is an open websocket with the AOI Server and if not then it establishes it. Then it issues a Client Application De-registration request, encoded as a JSON-RPC 2.0 [JSON-RPC] request object, with:

- “method” member set to “clientAppDeregistrationReq”
- “id” member including an unique identifier for each request
- “params” member including the following parameter:
    - “clientID” : mandatory, string
    - “clientAppToken” : mandatory, string

Example:

```
{
      "jsonrpc": "2.0",
      "method": " clientAppDeregistrationReq ",
      "id": "117",
      "params": {
         “clientID”: “aoi123456789”,
         “tokenSeed”: “token123456789”;
        }
}
```

Then the AOI Server returns an Client Application De-registration response encoded as a JSON-RPC 2.0 [JSON-RPC] response object, with:

- “id” member matching the identifier included in the request object
- “result” or “error” member, in case of success or error response respectively, including the following parameters:
    - “code”: mandatory, string
    - “message”: mandatory, string. Corresponds to the “desc” parameter

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "117",
    "result": {
          “code”: 200,
          “message”: “OK“,
          }
}
```

Example of error response:

```
{   "jsonrpc": "2.0",
    "id": "117",
    "error": {
          code: 503,
          message: “Service Unavailable“
          }
}
```

10.1.7 Notification Channel Maintenance

This section describes the keep-alive message request sent from AOI Client to AOI Server using AOI-1 interface to keep the Notification Channel open and the response to this request respectively. This is specifically for the case when the AOI Client is within Firewall and NAT networks and for WebSocket protocol binding.

In this case the notification channel may close after a certain idle time. To keep the channel open a keep-alive message should be sent from time to time encoded as a JSON-RPC 2.0 [JSON-RPC] request object, with the following elements:

- “method” member set to “keep-alive”
- “id” an identifier for the ‘keep-alive’ request

Example of a keep-alive request:

```
-->{
      "jsonrpc": "2.0",
      "method": "keep-alive",
      “id”: 1

}
```

Then AOI Server returns a response to keep-alive request encoded as a JSON-RPC 2.0 [JSON-RPC] request object, with:

- “result” member set to “200 OK” when the response is positive, i.e. successful.
- “id” member is included and should have the same value with the ‘id’ member in the ‘keep-alive’ request.

Example of success response to a keep-alive request:

```
<--{
      "jsonrpc": "2.0",
```

"result": "200 OK",
"id": 1

}

Note: The duration between two keep-alive transmissions, duration between two successive keep-alive retransmissions and the number of retransmissions before declaring that the channel is dead are implementation issues.

10.2 AOI-2 Interface

10.2.1 Notification Channel De-activation

When the AOI Server wants to deactivate the notification channel with the AOI Client, it will send a Notification Channel De-activation request, which could be encoded as, e.g. JSON-RPC 2.0 [JSON-RPC] request object, with:

- "method" member set to "ChannelDeactivationReq"
- "id" member including an unique identifier for each request
- "params" member including the following parameters:
  - "SeverID" : mandatory, string
  - "ChannelType": mandatory, string, =["pushOTA", "UDP", "TCP"]

Example:

{

"jsonrpc": "2.0",

"method": "ChannelDeactivationReq",

"id": "211",

"params": {

ServerID: "aoiServer000123",

ChannelType: "TCP"

}

}

The AOI Client returns a Notification Channel De-activation response, which could be encoded as, e.g. JSON-RPC 2.0 [JSON-RPC] response object, with:

- "id" member matching the identifier included in the request object
- "result" or "error" member, in case of success or error response respectively, including the following parameters:
  - "code": mandatory, string
  - "message": mandatory, string. Corresponds to the "desc" parameter

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "211",
    "result": {
        code: 200,
        message: “OK“
        }
}
```

Example of error response:

```
{   "jsonrpc": "2.0",
    "id": "211",
    "error": {
        code:400,
        message: “Bad Request“
        }
}
```

10.2.2 Notification Delivery

When the AOI Server wants to deliver a notification to the AOI Client, it will send a Notification Delivery request, which could be encoded as, e.g. JSON-RPC 2.0 [JSON-RPC] request object, with:

- “method” member set to “NotificationDeliveryReq”
- “params” member including the following parameters:
    - “Notification” : mandatory (1 or multiple), NotificationType, including the following parameters:
        - “ClientAppToken”: mandatory, string
        - “payload”: optional, Notification Payload Structure
        - “priority”: mandatory, enumeration

Example:

```
{
```

```
                    "jsonrpc": "2.0",
                    "method": "NotificationDeliveryReq",
                    "id": "221",
                    "params": {
                Notification: {ClientAppToken: “ClientApp0xyz123”,
                payload: “…”,
                priority: “2”}
        }
}
```

Upon successful notification delivery the AOI Client returns a Notification Delivery success response encoded as, e.g. JSON-RPC 2.0 [JSON-RPC] response object, with:

- “id” member matching the identifier included in the request object
- “result” member including the following parameters:
    - “code”: mandatory, string
    - “message”: mandatory, string. Corresponds to the “desc” parameter

Example of success response:

```
{   "jsonrpc": "2.0",
    "id": "221",
    "result": {
        code: 202,
        message: “Accepted“
        }
}
```

Upon unsuccessful notification delivery the AOI Client returns a Notification Delivery error response encoded as, e.g. JSON-RPC 2.0 [JSON-RPC] response object, with:

- “id” member matching the identifier included in the request object
- “error” member including the following parameters:
    - “code”: mandatory, string
    - “message”: mandatory, string. Corresponds to the “desc” parameter

Example of error response:

```
{   "jsonrpc": "2.0",
    "id": "221",
    "error": {
        code: 415,
        message: “Unsupported Media Type “
        }
}
```

11.Security

11.1 Server Application Authentication

When receiving a notification request from a Server Application, the AOI Server SHOULD authenticate the Server Application.

If AOI Server requires authentication, if the authentication fails, the AOI Server SHALL reject notification request from the Server Application. Even if the authentication succeeds, the AOI Server SHALL reject notification request from the Server Application (e.g. Server Application is registered with a blacklist, number of notifications is beyond the limitation, etc…)

AOI Server SHOULD support the following mechanisms for Server Application authentication based on Server Application provider credential. As a credential, either Server Application provider ID and password, or Server Application provider's certificate is used.

- When using ID and password as a credential

  AOI Server SHALL authenticate Server Application by either HTTP Basic Authentication or HTTP Digest Authentication.

  AOI Server SHALL check HTTP Authorization Header in each notification request to check if the header value matches with correct Server Application provider ID and Server Application provider password. If HTTP Basic Authentication is used, HTTPS MUST be utilized between AOI Server and a Server Application.

  NOTE: how to register Server Application provider ID and password with AOI Server is out of scope of this specification

- When using Server Application provider certificate as a credential

  AOI Server SHALL authenticate Server Application provider by TLS mutual authentication based on Server Application provider's certificate as a client certificate and AOI Server's certificate as a server certificate.

AOI Server MAY support lightweight application provider authentication (e,g, for trial applications or for all applications depending on deployment policy):

- AOI Server SHALL allow notification delivery from Server Application developed by unregistered application provider, if the Server Application provides valid clientAppToken
- It depends on deployment policy if there is a quota and/or validation period for an application developer using lightweight authentication.

11.2 Secure Exchange of Data

AOI Enabler SHALL support transport layer security (TLS) and/or application layer security including authentication and encryption to guarantee the message confidentiality and integrity.

12.Release Information

12.1 Supporting File Document Listing

Doc Ref	Permanent Document Reference	Description
Supporting File		

Table 47: Listing of Supporting Documents in AOI 1.0 Release

12.2 OMNA Considerations

There is no required OMNA registration for AOI 1.0.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-ER-AOI-V1_0-20170404-A	04 Apr 2017	Status changed to Approved by TP TP Ref # OMA-TP-2017-0023-INP_AOI-V1_0_ERP_for_Final_Approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 ERDEF for AOI 1.0 - Client Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-AOI-C-001-M	AOI Client	[AOI-ER]:MCF

Table 48: ERDEF for AOI 1.0 Client-side Requirements

B.2 ERDEF for AOI 1.0 - Server Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-AOI-S-001-M	AOI Server	[AOI-ER]:MSF
OMA-ERDEF-AOI-S-002-O	AOI Auxiliary Server	AOI-A-S-MSF

Table 49: ERDEF for AOI 1.0 Server-side Requirements

B.3 SCR for AOI Client

Item	Function	Reference	Requirement
AOI-C-001-M	managing the access from/to the Client Applications on the device related to the AOI Enabler functionalities	Section 6.3.1.1.1	
AOI-C-002-M	receiving registration/de-registration from Client Applications	Section 6.3.1.1.1	
AOI-C-003-M	sending notification subscription request to AOI Server	Section 6.3.1.1.1	
AOI-C-004-M	initiating Notification Channel setup with AOI Server	Section 6.3.1.1.1	
AOI-C-005-M	receiving notifications from AOI Server	Section 6.3.1.1.1	
AOI-C-006-M	receiving notification subscription from the Client Applications	Section 6.3.1.1.1	
AOI-C-007-M	delivering notifications to the Client Applications	Section 6.3.1.1.1	
AOI-C-008-M	indicating to Server Application when it is getting overloaded	Section 6.3.1.1.1	
AOI-C-009-M	managing Shared AOI Connectivity with AOI Server	Section 6.3.1.1.1	
AOI-C-010-M	managing (e.g., add, remove, update) information related to Client Application	Section 6.3.1.1.1	

Item	Function	Reference	Requirement
AOI-C-011-M	managing (e.g. query, update) different status information (e.g. registered, unregistered) related to Client Application	Section 6.3.1.1.1	
AOI-C-012-M	waking up Client Applications when needed	Section 6.3.1.1.1	
AOI-C-013-M	configuring the criteria for notification delivery (e.g. configure threshold of notification frequency)	Section 6.3.1.1.1	
AOI-C-014-M	reporting the statistics information of the AOI Client (e.g. connect/disconnect activities, connection history, time period, number of notifications, number of kB sent or received, upload/download speed)	Section 6.3.1.1.1	
AOI-C-015-O	alerting disaster/emergency notifications to the user (in case no Client Application Token is indicated in the notification)	Section 6.3.1.1.1	

B.4 SCR for AOI Server

Item	Function	Reference	Requirement
AOI-S-001-M	receiving registration/de-registration from Server Applications	Section 6.3.1.1.2	
AOI-S-002-M	receiving registration/de-registration from AOI Client	Section 6.3.1.1.2	
AOI-S-003-M	receiving notifications from Server Applications over secured connection	Section 6.3.1.1.2	
AOI-S-004-M	receiving notification subscription from the AOI Client	Section 6.3.1.1.2	
AOI-S-005-M	delivering notifications to the AOI Client, respecting the policies	Section 6.3.1.1.2	
AOI-S-006-M	controlling traffic for notification messages based on service provider policy, such as a threshold per application, selected applications or all applications	Section 6.3.1.1.2	
AOI-S-007-M	maintaining AOI Connectivity if at least one Client Application is registered with the AOI Client	Section 6.3.1.1.2	
AOI-S-008-M	managing (e.g., add, remove, update) information related to Client Applications and Server Applications in case the latter are registered	Section 6.3.1.1.2	

Item	Function	Reference	Requirement
AOI-S-009-M	managing (e.g. query, update) different status information (e.g. registered, unregistered) related to Client Application	Section 6.3.1.1.2	
AOI-S-010-M	handling authentication of AOI Client	Section 6.3.1.1.2	
AOI-S-011-M	handling authorization of Server Applications	Section 6.3.1.1.2	
AOI-S-012-M	configuring the criteria for notification delivery (e.g. configure threshold of notification frequency)	Section 6.3.1.1.2	
AOI-S-013-M	reporting and managing the statistics information of the AOI Server and AOI Clients(e.g. connect/disconnect activities, connection history, time period, number of notifications, number of kB sent or received, upload/download speed)	Section 6.3.1.1.2	
AOI-S-014-M	handling authorized Server Applications responsible for emergency situation	Section 6.3.1.1.2	
AOI-S-015-M	receiving lightweight messages from the AOI Client and delivering them to Server Applications	Section 6.3.1.1.2	

B.5 SCR for AOI Auxiliary Server

Item	Function	Reference	Requirement
AOI-A-S-001-O	located within the MNO's network and provides support to the AOI Server, when this latter component lies outside of the MNO's network, in all those features that require a direct contact with the AOI Client from within the MNO's network	Section 6.3.1.1.3	
AOI-A-S-002-O	may be also used as a deployment option even if the AOI Server is placed within the MNO's network	Section 6.3.1.1.3	

Appendix C. Some suggestions for AOI Enabler deployment (Informative)

C.1 Deployment of multiple AOI Servers

The operator should be able to deploy and operate multiple AOI Servers. The AOI Client may communicate with more than one AOI Servers.

For the purpose of network efficiency the AOI Client should have only one associated AOI Server at a time.

C.2 AOI Server load control

The Notification Service shall enable the AOI Server to provide services in stable condition and prevent the AOI Server from being overloaded. A possible deployment option is that when an AOI Server is overloaded, the AOI Server may reject the access from some AOI Clients with a specific overload server error to guarantee stable operation of the AOI Server. If the access to AOI Server is failed, the AOI Client may attempt to find to another AOI Server on the basis of the Notification Service information and policy that are given from the operator or return fail to the application so that the application can establish a direct connection with the server application.

C.3 AOI Client in a Device

For the purpose of network efficiency, a device compatible with OMA AOI Enabler shall not allow Client Application(s) from associating with multiple AOI Clients at the same time. For example, AOI Service provider ensures that the device compatible with OMA AOI Enabler has only one active AOI Client.

AOI Client may be integrated into the Operating System on the device, or integrated with Client Application, or installed independently, depending on the deployment policy.

Appendix D. Example of notification payload (Informative)

This appendix provides examples of notification payload over AOI interfaces.

D.1 Example of notification payload

D.1.1 Simple notification

The following is an example of payload to notify a simple message.

```
{

    "alertMsg" : "Message received from OMA" ,

}
```

D.1.2 Notification with sound

The following is an example of payload with specified sound and icon

```
{

  "alertMsg" : “system alert”,

  "iconName" : 1,

  "soundName" : "dingdong.wav"

}
```

D.1.3 Notification with button

The following is an example of payload containing alert object:

```
{

  "alertMsg" : {
        “body” : “you need check new message”,
        “button” : “OK”
  },

  "iconName" : “icon.png”,

  "soundName" : "dingdong.wav"

}
```

Appendix E. Notification Delivery State (Informative)

The AOI Enabler SHALL support notification delivery state flow control. Detailed State Flow Control steps and its diagrams for AOI Server, and AOI Client are provided in F.3 and F.4 respectively.

E.1 General Procedure

This is a general model of notification delivery procedure in AOI Enabler. Notification Delivery Lifecycle is a cycle which explains complete flow of a notification delivery procedure from Server Application to Client Application as shown in figure1.

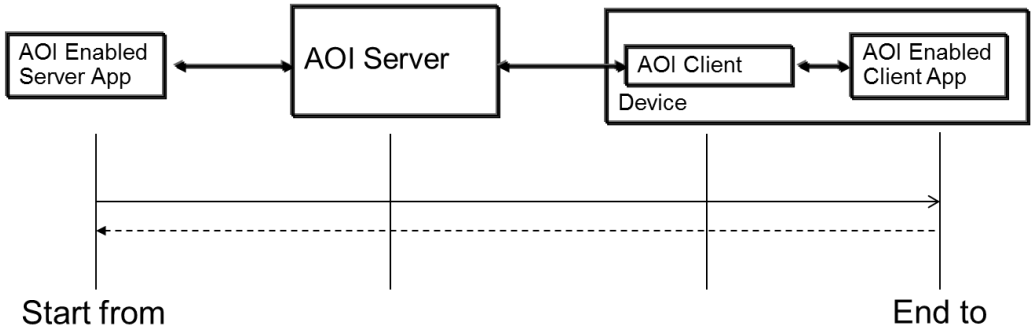

Figure 6: Notification Delivery Lifecycle

E.2 Notification Delivery State Flow Control

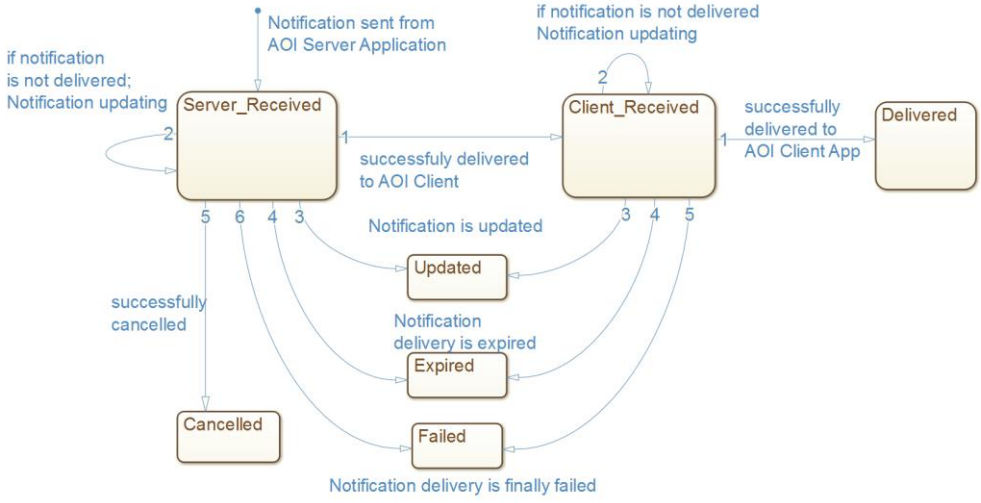

Figure 7: Notification Delivery State Flow Control

E.3 Notification Delivery State Flow Control for AOI Server

A state flow control diagram is depicted in Figure 7 for the AOI Server. There are at least 4 states in AOI notification delivery state transition on AOI Server, which are represented in solid ellipse in figure 1 including Failed, Server_Received, Updated, Cancelled, and Expired.

A solid arrow indicates the procedures in AOI for AOI Server. A triggering event to trigger the state transition is also noted on the arrow with the direction included.

Based on the Figure 7 above, the triggering events of each state transition are described respectively below:

- Server_Received: When the AOI Server Application start to send a notification, Notification Delivery is started as illustrated in Figure1. After the notification received by AOI Server, notification state successfully transits to Server_Received. If notification delivery to AOI client is temporary failed, notification state is not changed.
- Server_Received→Failed: When the server received notification is failed to deliver to AOI Client many times, notification delivery state transits from Server_Received to Failed. In this case, AOI Server informs the AOI Server Application through a feedback service why it failed.
- Server_Received→Updated: When the notification update request is successfully applied to the notification in AOI Server, notification delivery state transits from Server_Received to Updated. (Notification Updating on the AOI Server is only available during the server received notification is not delivered to AOI Client. If the server received notification is already delivered to AOI Client, AOI Server bypass the request to AOI Client)
- Server_Recevied→Expired: When the server received notification could not delivered to AOI Client before the specified the expiration time, notification delivery state transits from Server_Received to Expired. In this case, AOI Server informs the AOI Server Application through a feedback service when it expired.
- Server_Received→Cancelled: When the notification cancel request from AOI Server Application is accepted to AOI Server and the notification cancellation is successfully applied to the notification in AOI Server, notification delivery state transits from Server_Received to Cancelled. And then, AOI Server notifies the cancellation to the AOI Server Application. (Notification Cancellation on the AOI Server is only available during the server received notification is not delivered to AOI Client. If the server received notification is already delivered to AOI Client, AOI Server)

E.4 Notification Delivery State Flow Control for AOI Client

A state flow control diagram for AOI Client is depicted in Figure 7. There are 4 states in AOI notification delivery state transition on AOI Client, which are represented in solid ellipse in Figure 7 including Client_Received, Failed, Delivered, Cancelled, Expired, and Deleted.

- Client_Received: When the notification is delivered from AOI Server to AOI Client, Notification Delivery in AOI Client is started as illustrated in Figure2. If notification delivery to AOI Client Application is temporary failed, notification state is not changed.
- Clent_Received→Delivered: When the client received notification is successfully delivered to AOI Client Application, notification delivery state transits from Client_Received to Delivered.
- Client_Received→Failed: When the delivered notification in AOI Client is failed to deliver to AOI Client Application many times, notification delivery state transits from Client_Received to Failed. In this case, AOI Client informs the AOI Server Application through a feedback service why it failed.
- Client_Received→Expired: When the delivered notification in AOI Client could not delivered to AOI Client Application before the specified the expiration date, notification delivery state transits from Client_Received to Expired. In this case also, AOI Client informs the AOI Server Application through a feedback service when it expired.
- Client_Received→Updated: When the notification update request is successfully applied to the notification in AOI Client, notification delivery state transits from Server_Received to Updated. (Notification Updating on the AOI Client is only available during the client received notification is not delivered to AOI Client Application. If the client received notification is already delivered to AOI Client Application, AOI Client informs the AOI Server Application through a feedback service why update failed)

E.5 Notification Delivery State Check and Indication

After sending the notification to the AOI Server, the Server Application can check the delivery state of this notification and the Server Application will be aware whether the notification(s) is(are) delivered to the AOI Client or the Client Application. The Server Application also can require to be informed when the notification delivery state is changed. So the Server Application will be informed when the notification(s) is(are) delivered to the AOI Client or the Client Application.

Appendix F. Flows(Informative)

F.1 AOI Client Registration

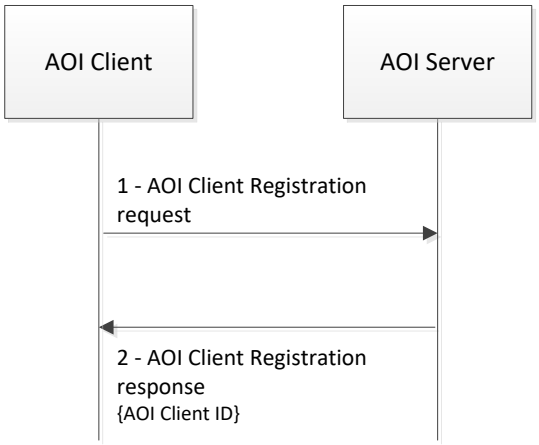

Figure 8: AOI Client Regsitration

F.2 Notification Channel Activation

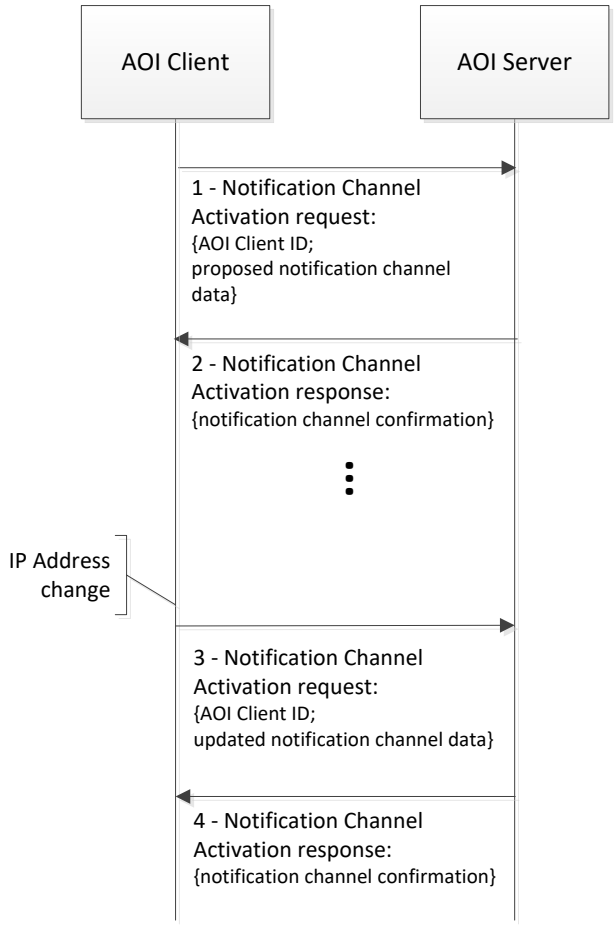

Figure 9: Notification Channel Activation without Auxiliary Server

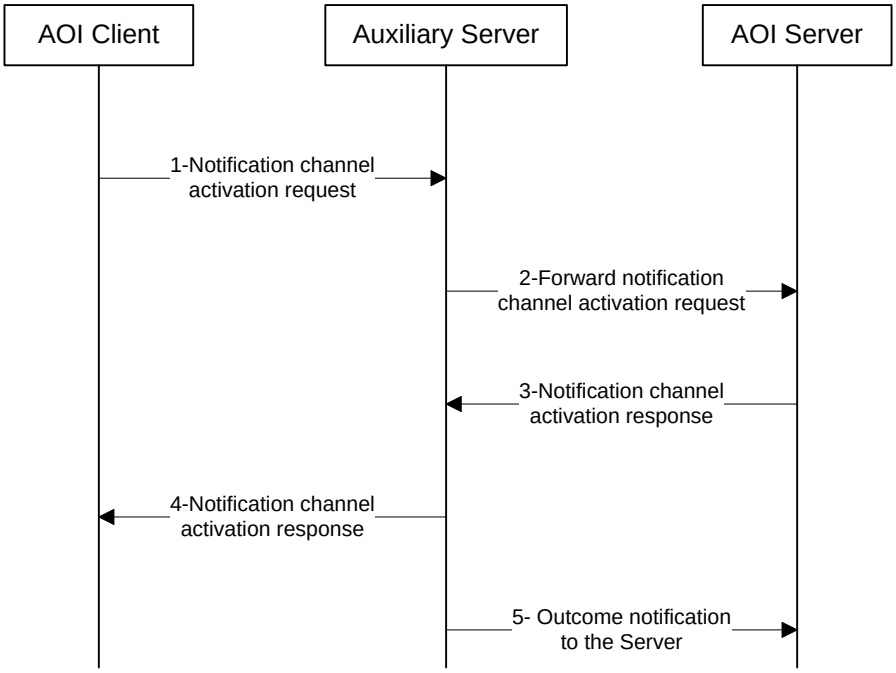

Figure 10: Notification Channel Activation with Auxiliary Server

F.3 Client Application Registration

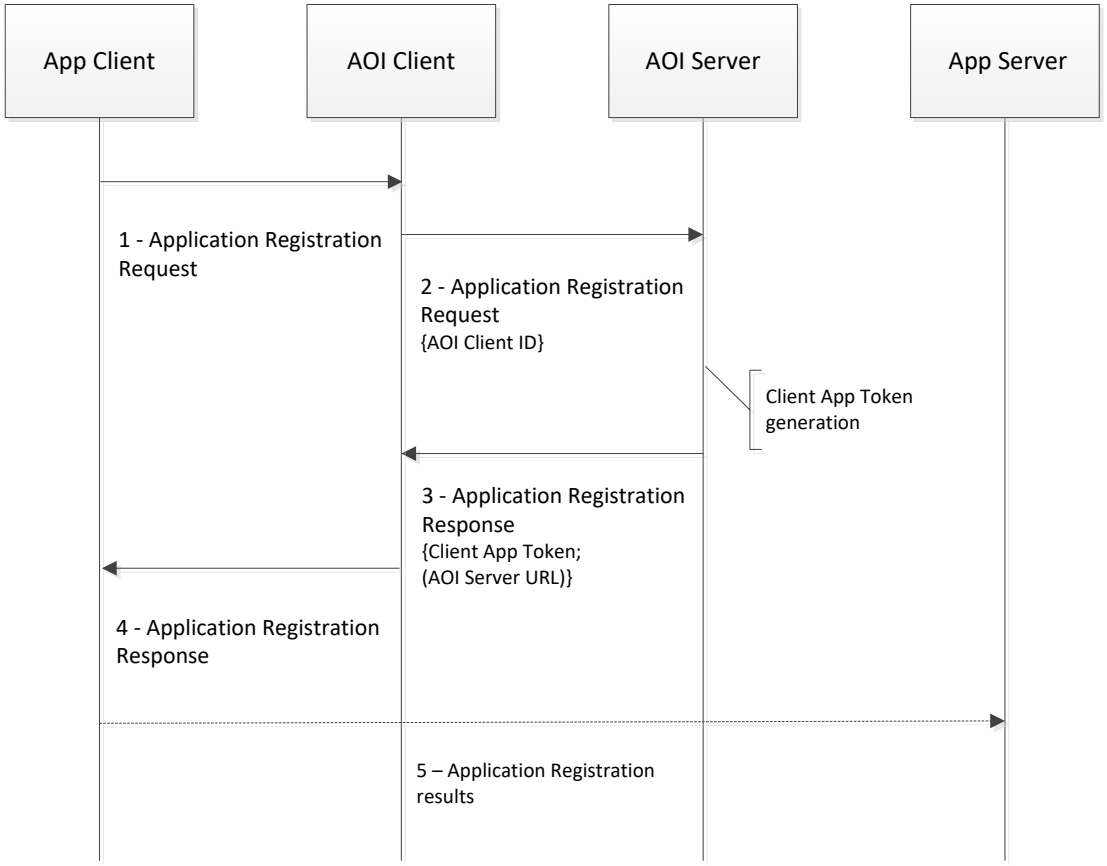

Figure 11: Client Application Registration

F.4 Client Application De-registration

F.4.1 Client Application De-registration(notification un-subscription)

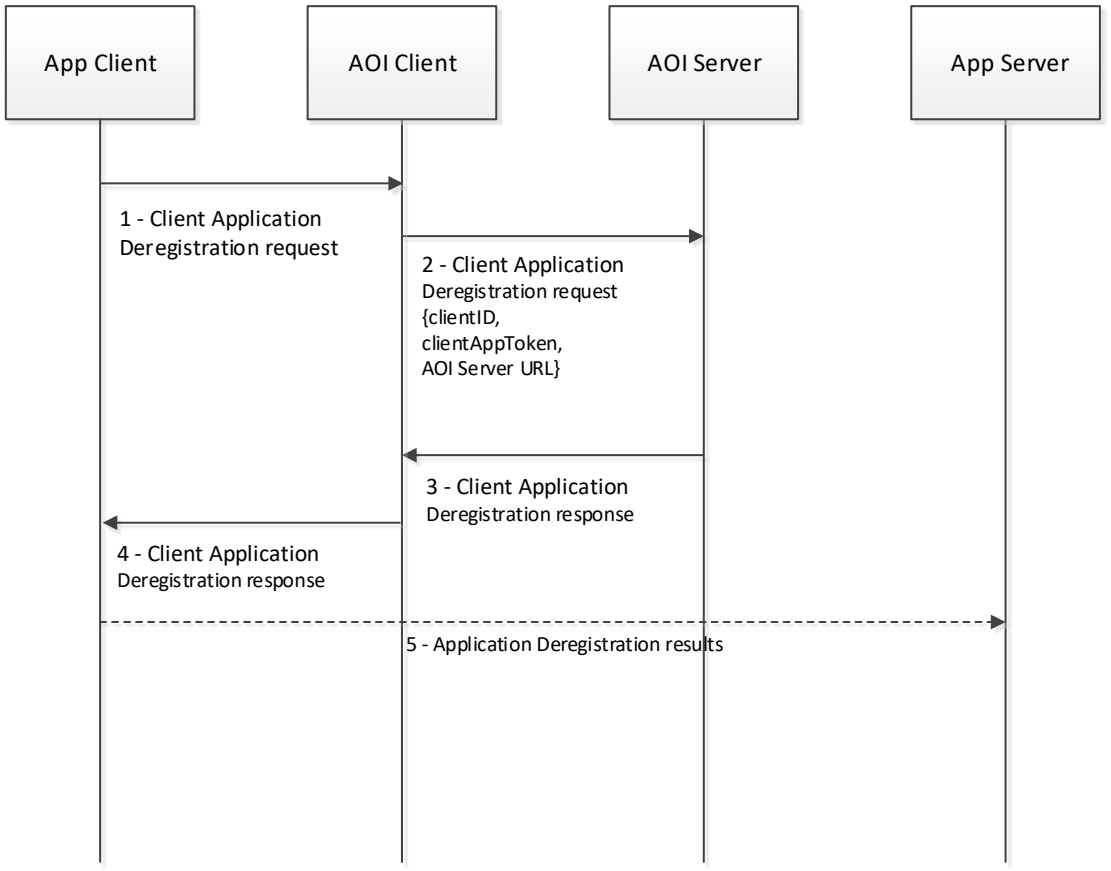

Figure 12: Client Application De-registration

F.4.2 Client Application De-registration(Client App un-installation)

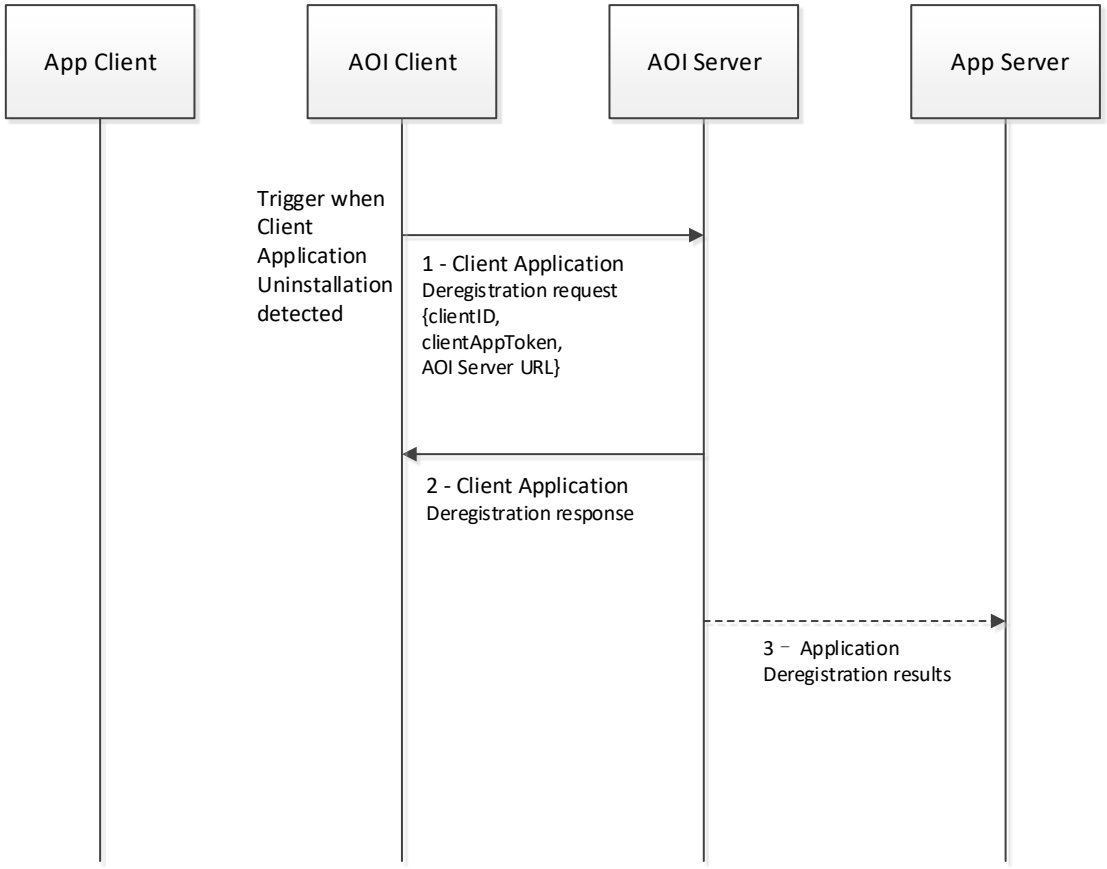

Figure 13: Client Application De-registration

F.5 Notification Delivery

F.5.1 Notification Delivery (Normal Flow)

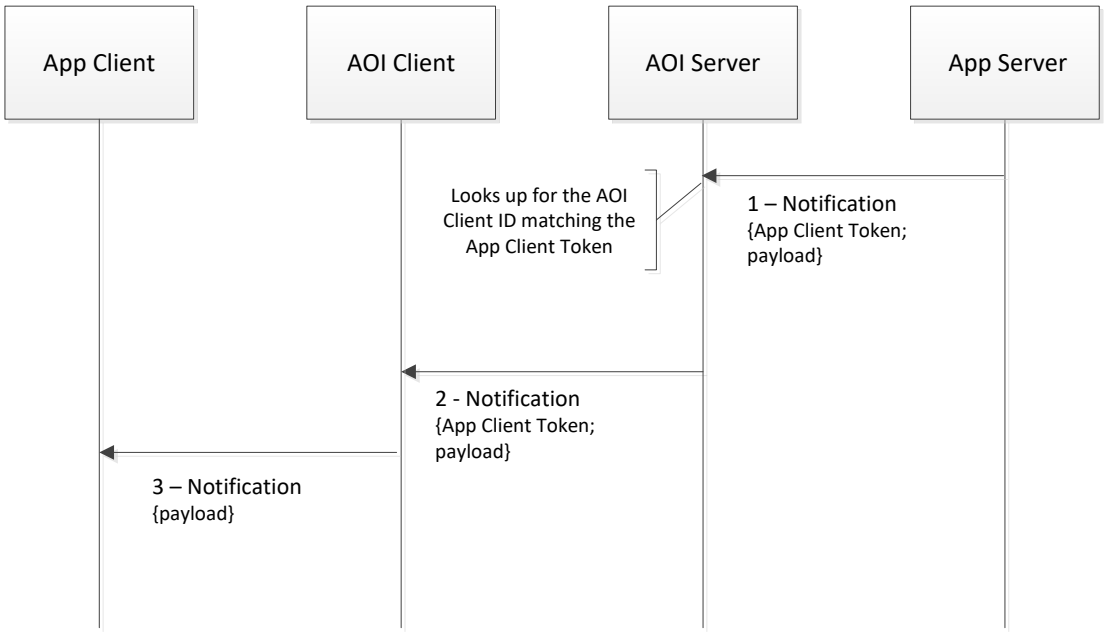

Figure 14: Notification Delivery

F.5.2 Notification Delivery (De-Register)

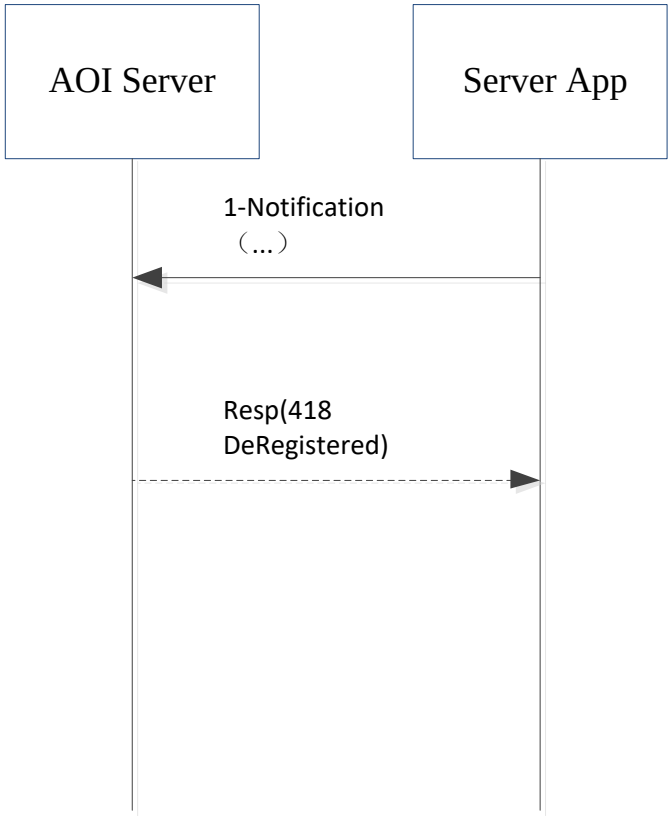

Figure 15: Notification Delivery of De-registration

F.5.3 Notification Delivery (No channel is available)

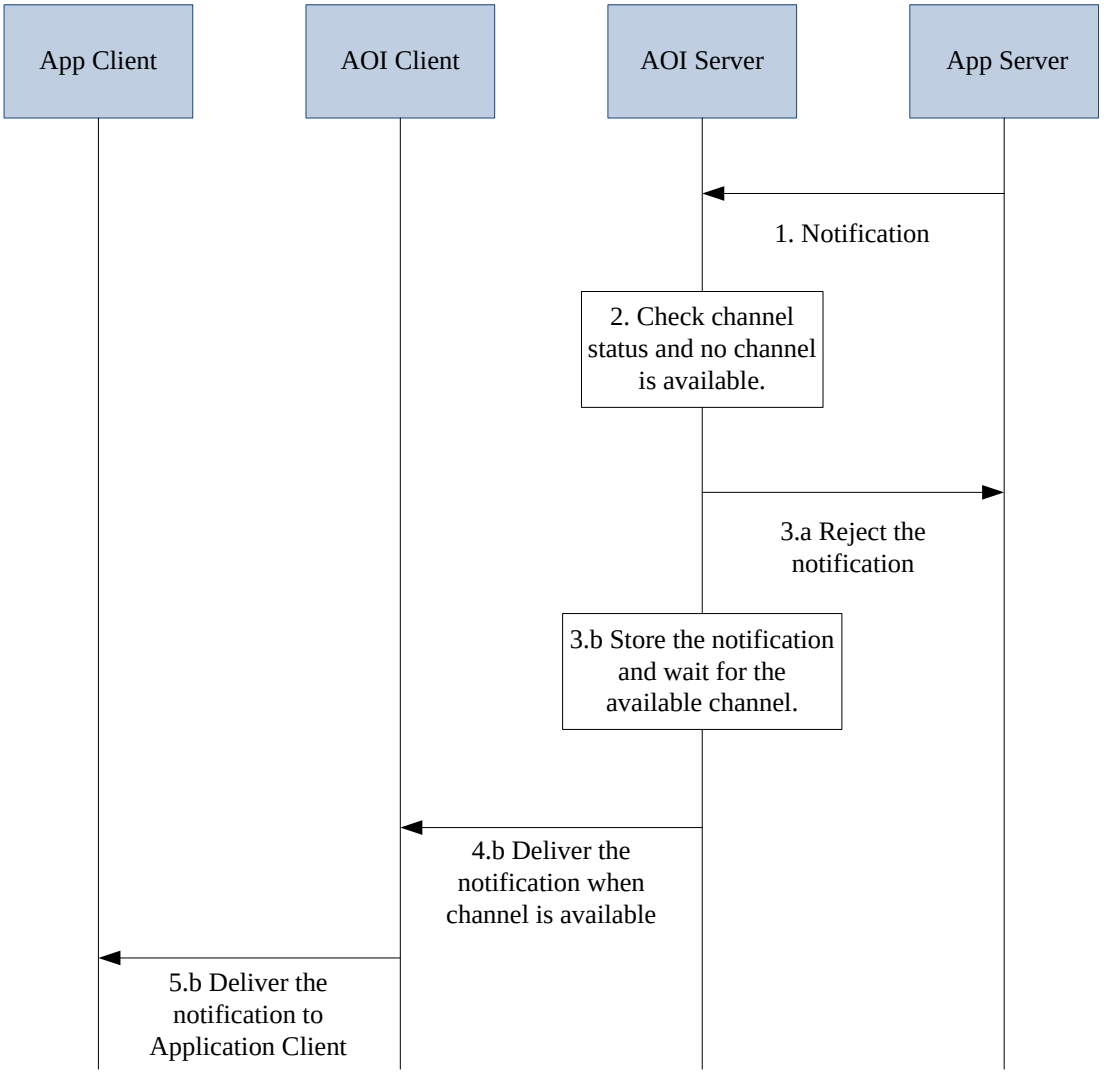

Figure 16: Notification Delivery Based on Channel Status

F.5.4 Notification Delivery (with priority)

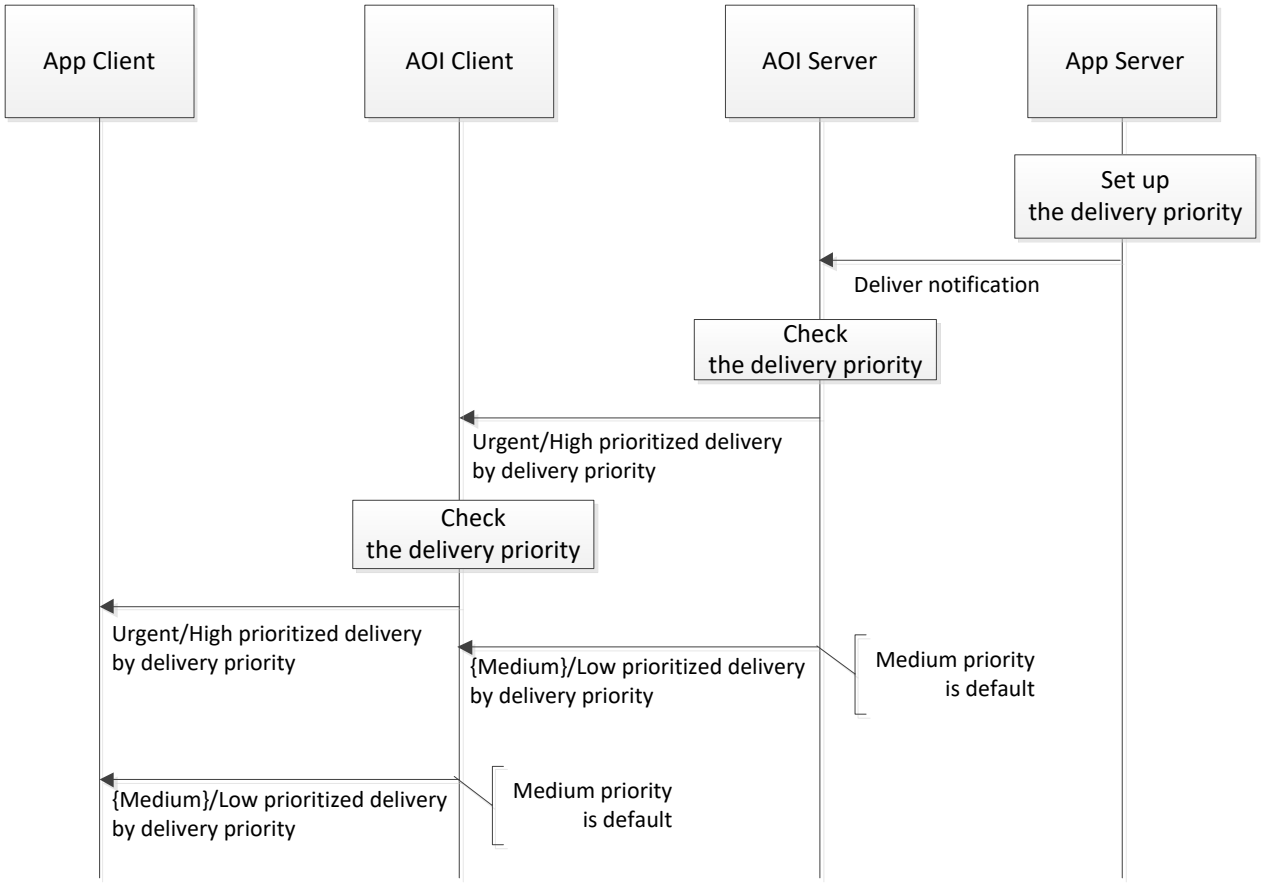

Figure 17: Notification Delivery Based on Priority

F.5.5 Notification Delivery Expiration

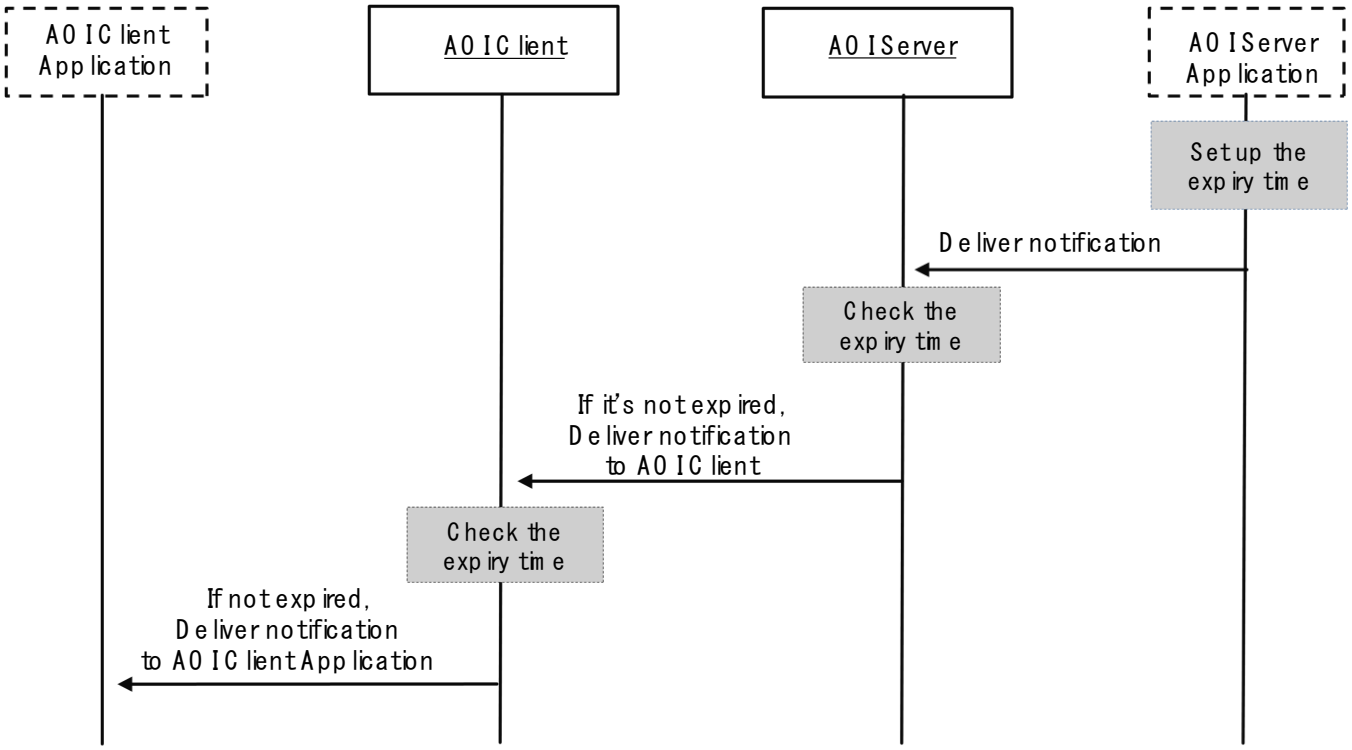

Figure 18: Notification Delivery Expiration

F.5.6 Notification Channel Switching

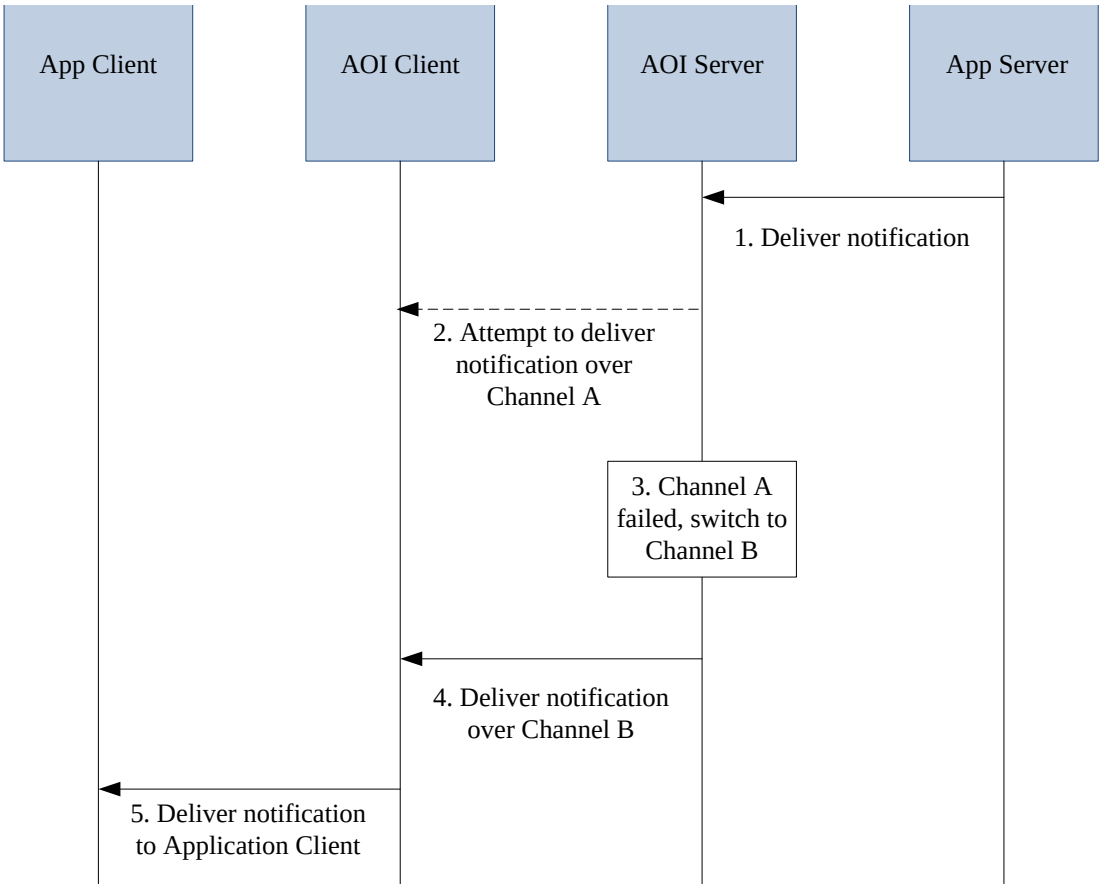

Figure 19: Notification Channel Switching

F.5.7 Notification Delivery To Multiple Client Application

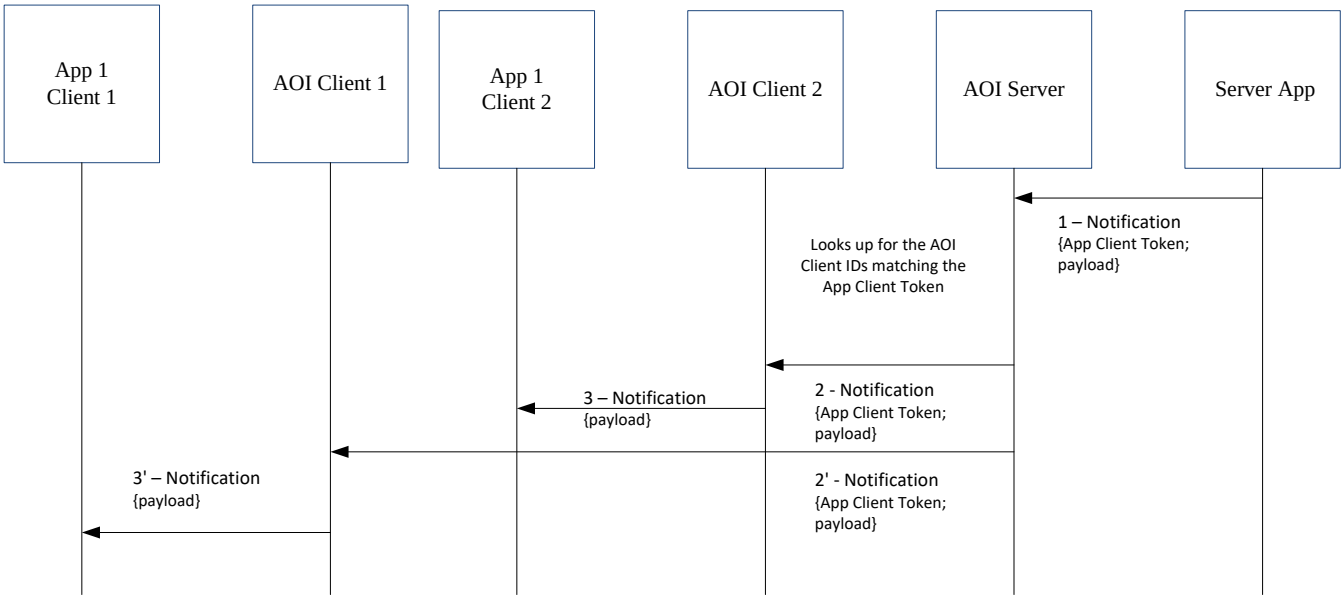

Figure 20: Notification Delivery to Multiple Client Applications

F.5.8 Notification Delivery To All Client Applications

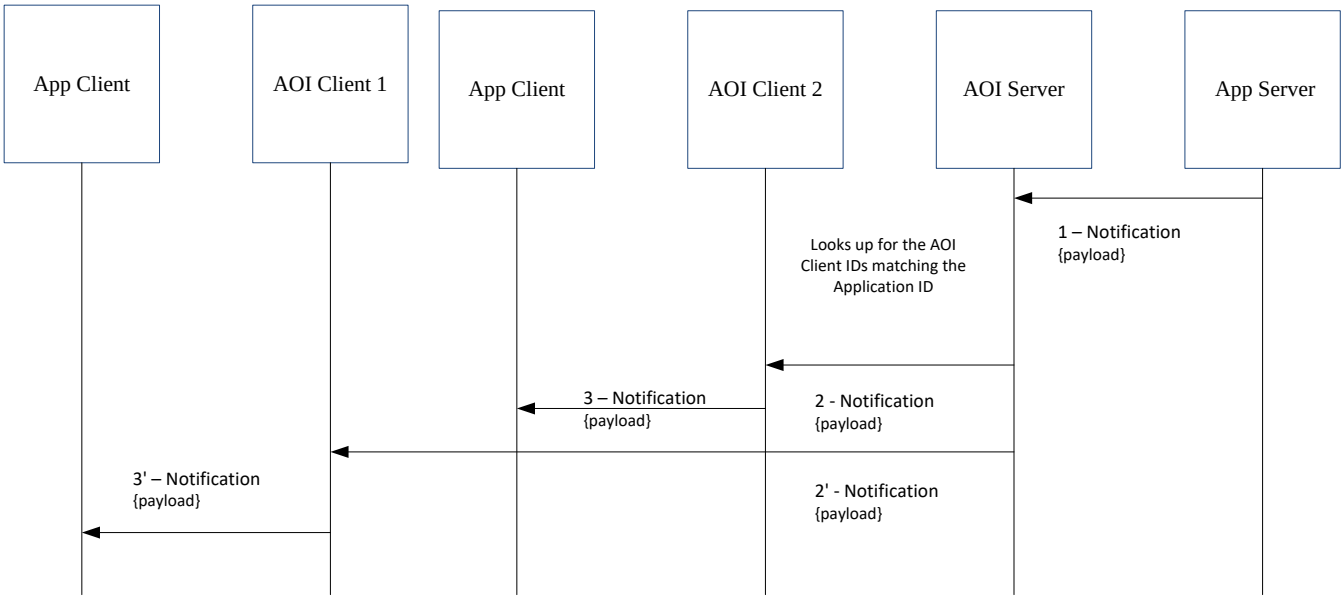

Figure 21: Notification Delivery to All Client Apps Per application

F.5.9 Notification payload size limitation

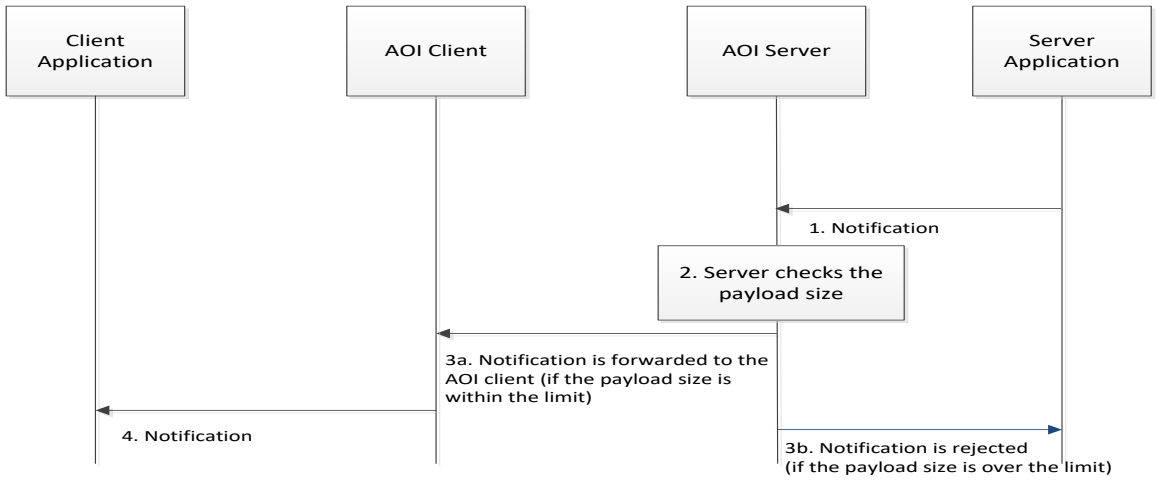

Figure 22: Call flow for Notification payload size limitation

F.5.10 Client Application Wake-up

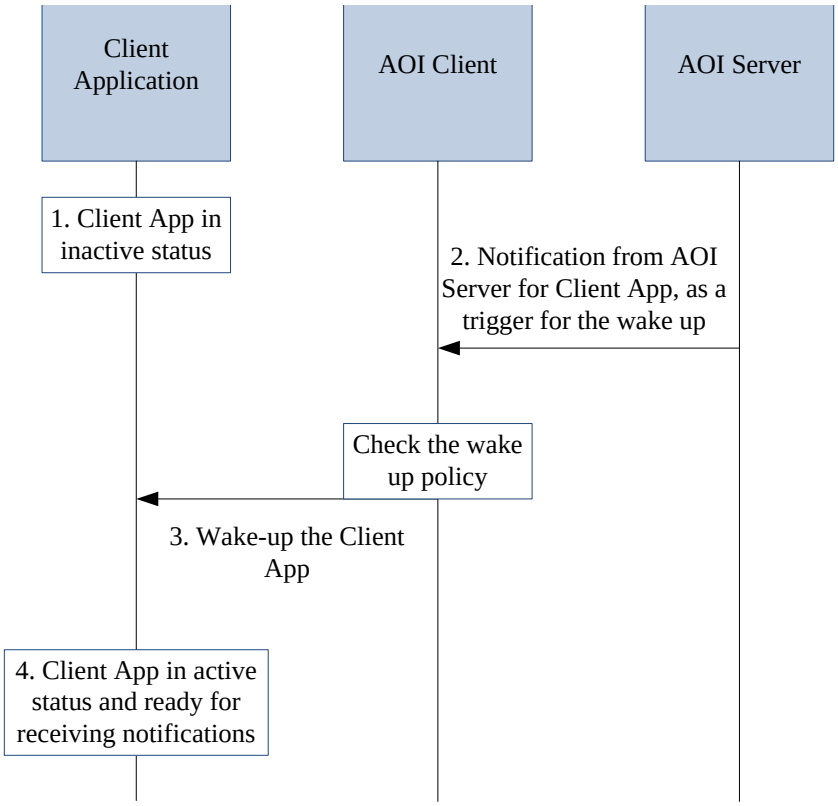

Figure 23: Client Application Wake-up

F.6 AOI Client Wake-up

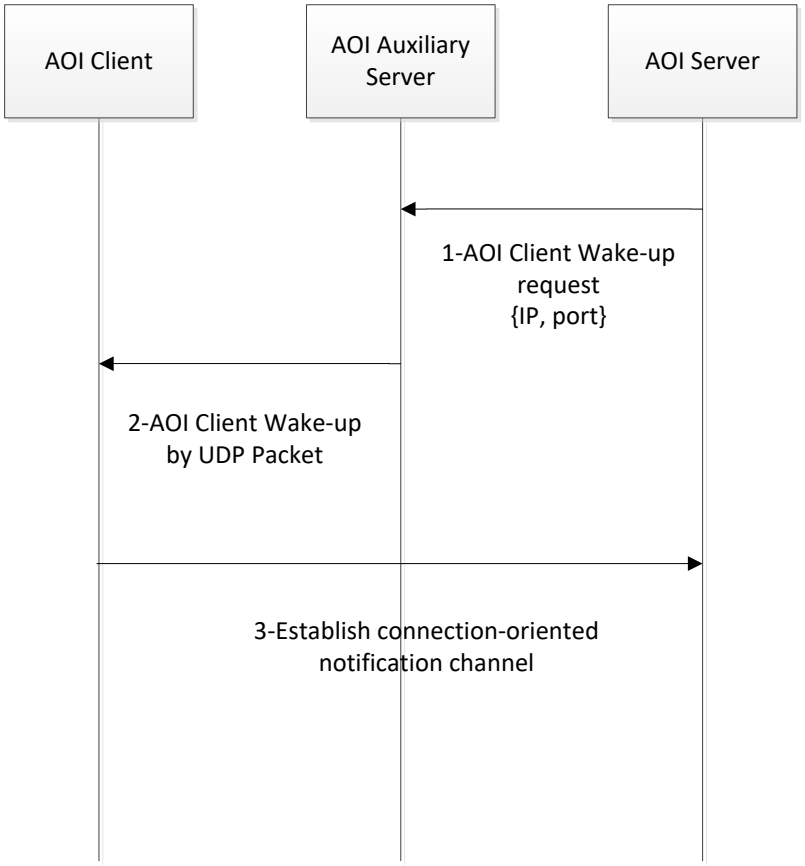

Figure 24: AOI Client Wake-up

F.7 Notification Channel De-activation by AOI Client

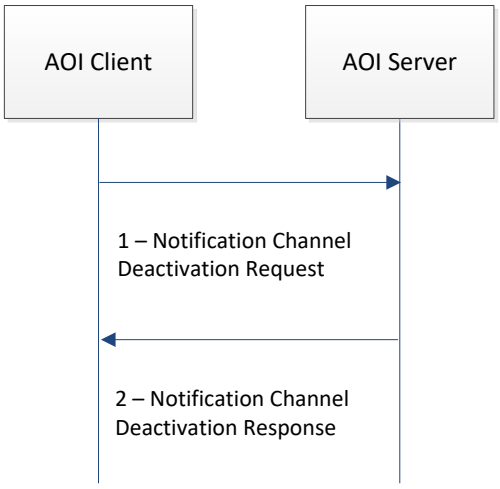

Figure 25: Notification Channel De-activation by AOI Client

F.8 Notification Channel De-activation by AOI Server

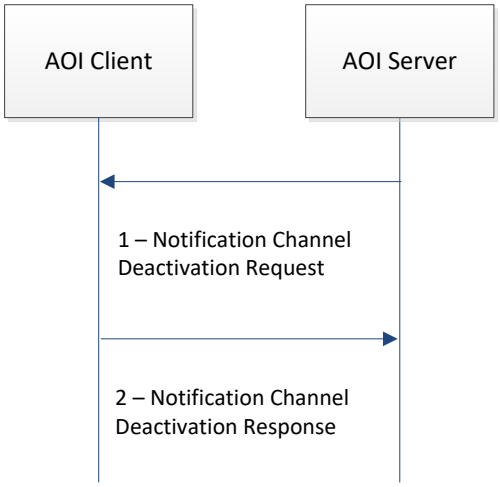

Figure 26: Notification Channel De-activation by AOI Server

F.9 Notification Channel Maintenance

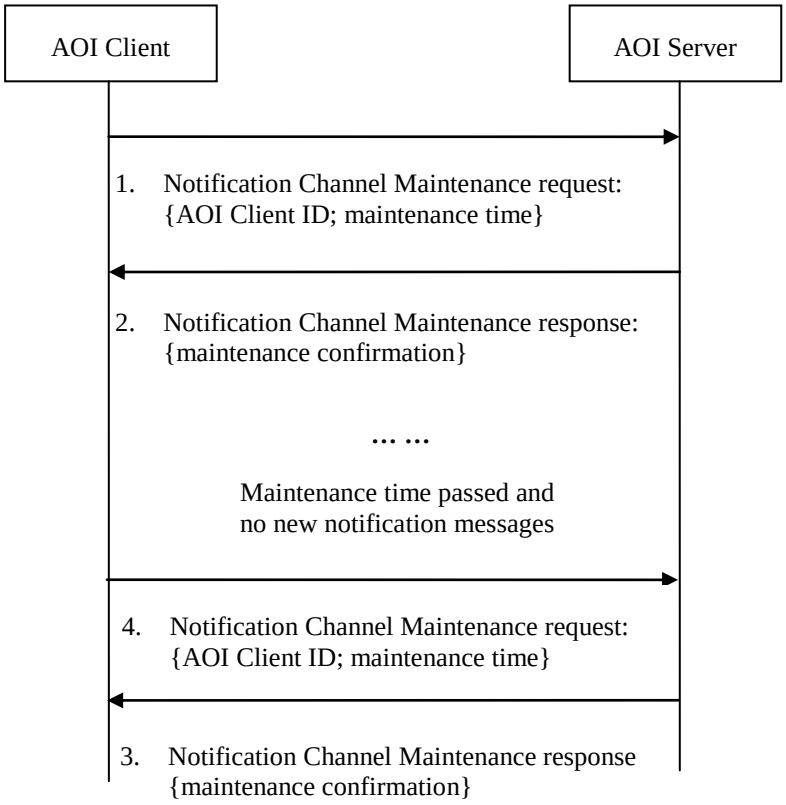

Figure 27: Notification Channel Maintenance

F.10 AOI Client De-registration

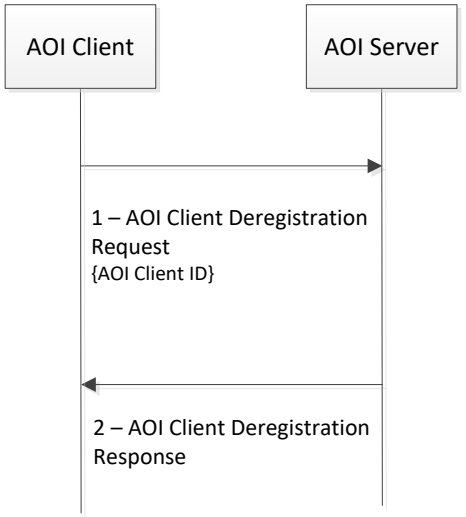

Figure 28: Active AOI Client De-registration

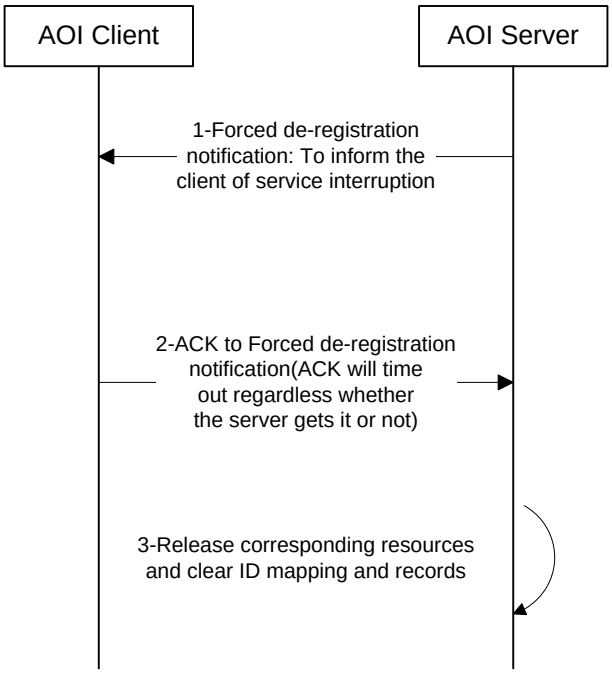

Figure 29: Forced AOI Client De-registration

F.11 Server Application Registration

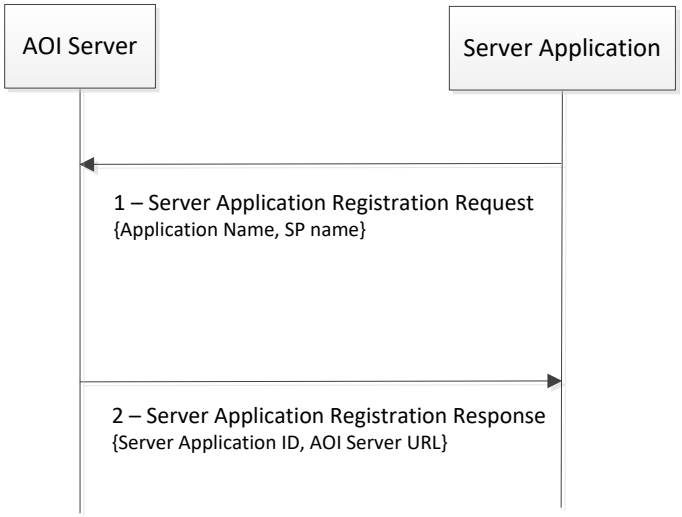

Figure 30: Server Application Registration flow

F.12 Server Application De-registration(Server Application triggered)

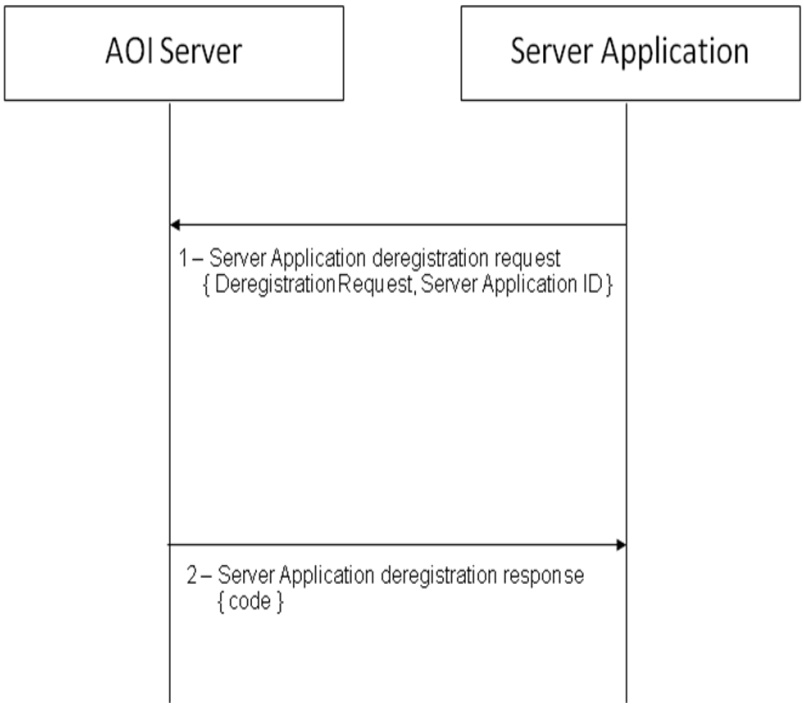

Figure 31: Server Application de-registration flow

F.13 Server Application De-registration (AOI Server triggered)

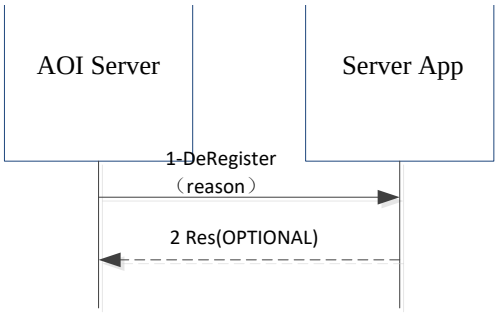

Figure 32: Server Application de-registration flow (AOI Server triggered)

F.14 Upward Lightweight Messaging

The following figure is given to illustrate the flow and procedures of upward lightweight messaging delivery from the Client Application to the Server Application over the dedicated connection in AOI system.

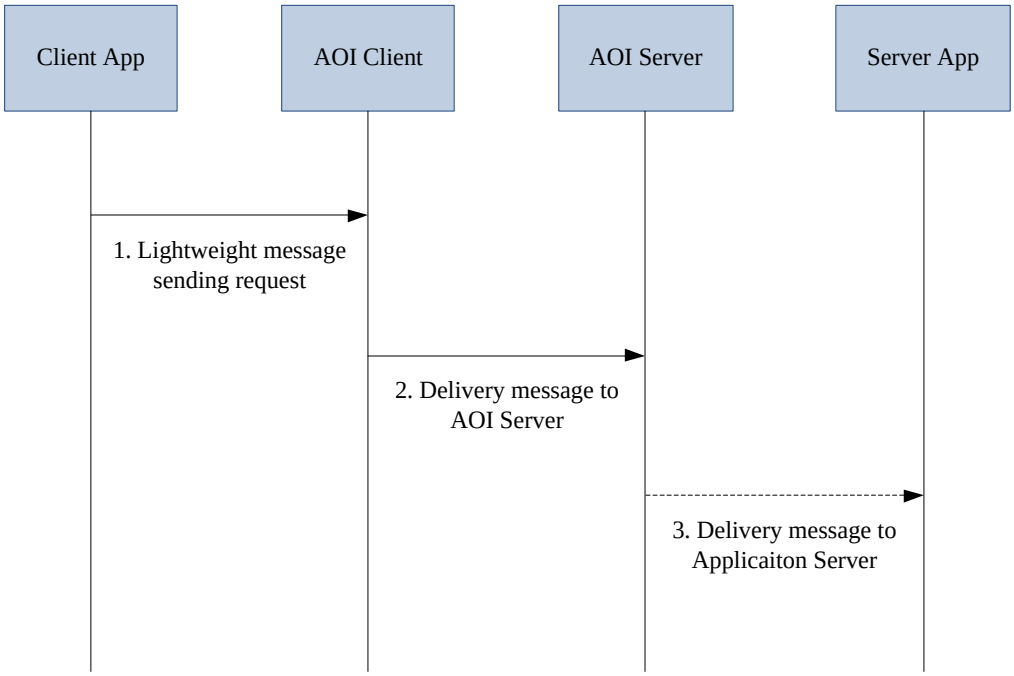

Figure 33: Upward Lightweight Messaging

Note: The upward lightweight messaging mechanism is OPTIONAL in OMA AOI Enabler. The AOI Service provider has the choice to deploy or not to deploy this mechanism.

Appendix G. Notification Payload Structure (Informative)

Each notification includes a payload. The payload contains information about how the system should notify the user as well as any custom data you provide. The maximum size allowed is subject to the policies set by service providers; if notification exceeds this limit, AOI Server SHALL refuse that.

For each notification, compose a JSON structure (as defined by RFC 4627). This JSON object must contain another JSON structure identified by the key NPS (Notification Payload Structure). The NPS structure contains one or more properties that specify the following actions:

- An alert message to display to the user
- A number to badge the application icon with
- A sound to play
- Extended data for notification based processing

The following table describes the elements of the Notification Payload Structure:

Element	Type	Cardinality	Description
alertMsg	String (or Alert JSON structure)	1	This element contains the alert text information. If this property is included, the system displays a standard alert. You may specify a string as the value of alert or a JSON structure as its value. If you specify a string, it becomes the message text of an alert with two buttons: Close and View. If the user taps View, the application is launched.
iconName	String	0…1	This element specifies the name of an icon file to display as the application icon. If this property is absent, the icon is not changed. To remove the icon, set the value of this property to 0.
soundName	String	0…1	This element specifies the name of a sound file. The sound in this file is played as an alert. If the sound file doesn't exist or default is specified as the value, the default alert sound is played. The audio must be in one of the audio data formats that are compatible with system sounds;
Extension	String (or JSON structure)	0…1	This element contains the extended data for notification based processing.

Table 50: Notification Payload Structure

The following table describes the elements of the alertMsg JSON structure

Element	Type	Cardinality	Description
Body	string	1	This element contains the text of the alert message.
Button	String	0…1	This element contains the text of buttons in the alert window. If a string is specified, the system displays an alert with specified text buttons.

Table 51: Alert JSON structure