App Note: Notification

05/23/2017

Introduction

Device Notifications and App Notifications are similar to property triggers and application triggers except they pertain to a device instead of a property. Typical usage includes being notified via email, SMS, or push notifications when a device loses or connects with the Ayla Cloud service. The Device Notification is used to set the condition to fire the associated App Notification.

Notification can be used on per device basis or set using templates from UI. When a device is unregistered, device notifications and their child app notifications are deleted if they were created programmatically. Those created via templates are retained.

Properties can have one or more associated triggers. Property Triggers are used to launch an action when a specified condition is met on the device. When the trigger fires, it launches an application as it is defined in an associated application trigger. Typical applications that can be triggered include: email, push, and SMS notification.

AylaDeviceNotification

These are all the methods that can be called on AylaDevice Object in the SDK relating to DeviceNotifications

1. **Create a Notification**
   Creates an AylaDeviceNotification in the Cloud

```
createNotification(final AylaDeviceNotification
notification,final Response.Listener<AylaDeviceNotification>
successListener,final ErrorListener errorListener)
```

params:
AylaDeviceNotification notification: A new AylaDeviceNotification object with all the desired values set
Response.Listener<AylaDeviceNotification> successListener : A Listener to Receive on successful Creation of Notification
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

2. **Update a Notification**
   Updates an existing AylaDeviceNotification in the Cloud

```
updateNotification(final AylaDeviceNotification
notification,final Response.Listener<AylaDeviceNotification>
successListener,final ErrorListener errorListener)
```

params:
AylaDeviceNotification notification: An update AylaDeviceNotification object with all the desired values set
Response.Listener<AylaDeviceNotification> successListener : A Listener to Receive on successful Update of Notification
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

3. **Fetch a Notification**
   Fetches an Array of AylaDeviceNotifications from the Cloud

```
fetchNotifications(final
Response.Listener<AylaDeviceNotification[]>
successListener,final ErrorListener errorListener)
```

params:
Response.Listener<AylaDeviceNotification[]> successListener : A Listener to Receive on successful fetch of DeviceNotifications
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

4. **Delete a Notification**
   Deletes an existing AylaDeviceNotification from the Cloud

```
deleteNotification(final AylaDeviceNotification
notification,final Response.Listener<EmptyResponse >
successListener,final ErrorListener errorListener)
```

params:
AylaDeviceNotification notification:An existing AylaDeviceNotification fetched from the Cloud
Response.Listener<EmptyResponse> successListener : A Listener to Receive on successful Deletion of Notification

ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

AylaDeviceNotificationApp

These are all the methods that can be called on AylaDeviceNotificationApp Object

1. **Create App**

Creates an AylaDeviceNotificationApp in the Cloud

```
createApp(final AylaDeviceNotificationApp notificationApp,final
Response.Listener<AylaDeviceNotificationApp>
successListener,final ErrorListener errorListener)
```

params:

AylaDeviceNotificationApp notificationApp: A new AylaDeviceNotificationApp object with all the desired values set
Response.Listener<AylaDeviceNotificationApp> successListener : A Listener to Receive on successful Creation of NotificationApp
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

2. **Update App**

Updates an existing AylaDeviceNotificationApp in the Cloud

```
updateApp(final AylaDeviceNotificationApp notificationApp,final
Response.Listener<AylaDeviceNotificationApp>
successListener,final ErrorListener errorListener)
```

params:

AylaDeviceNotificationApp notificationApp: An updated AylaDeviceNotificationApp object with all the desired values set
Response.Listener<AylaDeviceNotificationApp> successListener : A Listener to Receive on successful update of NotificationApp
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

3. **Fetch Apps**

Fetches an array of existing AylaDeviceNotificationApps from the Cloud

fetchApps(final Response.Listener<AylaDeviceNotificationApp[]> successListener,final ErrorListener errorListener)

params:

Response.Listener<AylaDeviceNotificationApp[]> successListener : A Listener to Receive on successful fetch of AylaDeviceNotificationApps

ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

4. **Delete App**

Deletes an existing AylaDeviceNotificationApp in the Cloud

deleteApp(final AylaDeviceNotificationApp notificationApp,final Response.Listener<AylaDeviceNotificationApp> successListener,final ErrorListener errorListener)

params:

AylaDeviceNotificationApp notificationApp: An existing AylaDeviceNotificationApp fetched from Cloud

Response.Listener<AylaDeviceNotificationApp> successListener : A Listener to Receive on deletion of NotificationApp

ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

Common Setters and Getters for AylaDeviceNotification

```
public NotificationType getNotificationType();
public String getDeviceNickname();
public Integer getThreshold();
public String getUrl();
public String getMessage();

public void setNotificationType(NotificationType notifType);
public void setDeviceNickname(String deviceNickname);
public void setThreshold(Integer threshold);
```

```
public void setUrl(String url);
public void setMessage(String message);
```

```
NotificationType is an enum as defined below
public enum NotificationType {
    OnConnect("on_connect"),
    IPChange("ip_change"),
    ConnectionLost("on_connection_lost"),
    ConnectionRestore("on_connection_restore");
}
```

Common Usage of AylaDeviceNotification

If there are no existing preconfigured DeviceNotifications then create new DeviceNotification by calling createNotification as follows

```
AylaDeviceNotification deviceNotification = new
AylaDeviceNotification();
deviceNotification.setNotificationType(notificationType);//This
could be SMS,email or Push
deviceNotification.setThreshold(thresholdValue);
deviceNotification.setMessage(msgText);
device.createNotification(deviceNotification, new
Response.Listener<AylaDeviceNotification>() {
    @Override
    public void onResponse(AylaDeviceNotification response) {
     //Created AylaDeviceNotification is returned
    }
}, new ErrorListener() {
    @Override
    public void onErrorResponse(AylaError error) {
//Log this error or Show a Toast message
    }
});
```

In case of already existing Notification call the fetch method as follows

```
 device.fetchNotifications(new
Response.Listener<AylaDeviceNotification[]>() {
    @Override
    public void onResponse(AylaDeviceNotification[] response) {
     //Iterate through the Array and select the
```

```
AylaDeviceNotification that needs an update
    }
}, new ErrorListener() {
    @Override
    public void onErrorResponse(AylaError error) {
//Log this error or Show a Toast message
    }
});
```

Once the `AylaDeviceNotification` is fetched from the Cloud create an app as follows

```
AylaDeviceNotificationApp app = new AylaDeviceNotificationApp();
//In case of email
AylaEmailTemplate template = new AylaEmailTemplate();
template.setEmailSubject(notificationEmailSubject);
template.setEmailTemplateId(notificationEmailTemplateId);
template.setEmailBodyHtml(notificationEmailBodyHTML);
app.configureAsEmail(contact, message, null, template);
//In case of SMS
app.configureAsSMS(contact, message);
//In case of Push
app.configureAsPushAndroid(PushNotification.registrationId,
message, null, null);

deviceNotification.createApp(app, new
Response.Listener<AylaDeviceNotificationApp>() {
    @Override
    public void onResponse(AylaDeviceNotificationApp response) {
     //The newly created app is returned
    }
}, new ErrorListener() {
    @Override
    public void onErrorResponse(AylaError error) {
   //Log this error or Show a Toast message
  }

});
```

In case where AylaDeviceNotificationApp already exists call fetchApps first to retrieve the app then call updateApp to update the existing app.

AylaPropertyTrigger

Following methods can be called on AylaProperty that are related to Triggers

1. **Create a Trigger**
   Creates a Trigger for the Property

```
createTrigger(final AylaPropertyTrigger propertyTrigger,final
Response.Listener<AylaPropertyTrigger> successListener,final
ErrorListener errorListener)
```

params:

```
propertyTrigger: A new AylaPropertyTrigger object with all the
desired values set
Response.Listener<AylaPropertyTrigger> successListener : A
Listener to Receive on successful Creation of Trigger
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest
```

2. **Update a Trigger**
   Updates an existing AylaPropertyTrigger in the Cloud

```
updateTrigger(final AylaPropertyTrigger  propertyTrigger,final
Response.Listener<AylaPropertyTrigger> successListener,final
ErrorListener errorListener)
```

params:

```
propertyTrigger: An update AylaPropertyTrigger object with all
the desired values set
Response.Listener<AylaPropertyTrigger> successListener : A
Listener to Receive on successful Update of Trigger
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest
```

3. **Fetch a Trigger**
   Fetches an Array of AylaPropertyTriggers from the Cloud

```
fetchTriggers(final Response.Listener<AylaPropertyTrigger[]>
```

successListener,final ErrorListener errorListener)

params:
Response.Listener<AylaPropertyTrigger[]> successListener : A Listener to Receive on successful fetch of DeviceTriggers
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

4. **Delete a Trigger**
   Deletes an existing AylaPropertyTrigger from the Cloud

deleteTrigger(final AylaPropertyTrigger propertyTrigger,final Response.Listener<EmptyResponse > successListener,final ErrorListener errorListener)

params:
propertyTrigger:An existing AylaPropertyTrigger fetched from the Cloud
Response.Listener<EmptyResponse> successListener : A Listener to Receive on successful Deletion of Trigger
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

AylaPropertyTriggerApp

Following methods can be called on AylaPropertyTrigger related to Apps

1. **Create App**

Creates an TriggerApp in the Cloud

createApp(final AylaPropertyTriggerApp triggerApp,final Response.Listener<AylaPropertyTriggerApp> successListener,final ErrorListener errorListener)

params:
AylaPropertyTriggerApp triggerApp: A new AylaPropertyTriggerApp object with all the desired values set
Response.Listener<AylaPropertyTriggerApp> successListener : A

Listener to Receive on successful Creation of TriggerApp
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

2. **Update App**
Updates an existing AylaPropertyTriggerApp in the Cloud

updateApp(final AylaPropertyTriggerApp triggerApp,final Response.Listener<AylaPropertyTriggerApp> successListener,final ErrorListener errorListener)

params:
AylaPropertyTriggerApp triggerApp: An updated AylaPropertyTriggerApp object with all the desired values set
Response.Listener<AylaPropertyTriggerApp> successListener : A Listener to Receive on successful update of TriggerApp
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

3. **Fetch Apps**
Fetches an array of existing AylaTriggerApps from the Cloud

fetchApps(final Response.Listener<AylaPropertyTriggerApp[]> successListener,final ErrorListener errorListener)

params:
Response.Listener<AylaPropertyTriggerApp[]> successListener : A Listener to Receive on successful fetch of TriggerApps
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

4. **Delete App**
Deletes an existing AylaPropertyTriggerApp in the Cloud

deleteApp(final AylaPropertyTriggerApp triggerApp,final Response.Listener<AylaPropertyTriggerApp> successListener,final ErrorListener errorListener)

params:
AylaPropertyTriggerApp triggerApp: An existing AylaPropertyTriggerApp fetched from Cloud
Response.Listener<AylaPropertyTriggerApp> successListener : A Listener to Receive on deletion of TriggerApp
ErrorListener errorListener: An Error Listener should one occur.

This method returns an AylaAPIRequest

Common Setters and Getters for AylaPropertyTrigger

```
public String getTriggerType();
public String getCompareType();
public String getValue();
public String getPropertyNickname();
public String getDeviceNickname();
public String getRetrievedAt();
public Boolean getActive();
public String getPeriod();
public String getBaseType();
public String getTriggeredAt();

public void setActive(Boolean active);
public void setTriggerType(String triggerType);
public void setCompareType(String compareType);
public void setValue(String value);
public void setPropertyNickname(String propertyNickname);
public void setDeviceNickname(String nickname);
```

Common Usage of AylaPropertyTrigger

If there are no existing preconfigured PropertyTriggers then create new Trigger by calling createTrigger as follows

```
aylaProperty.createTrigger(trigger, new Response.Listener<AylaPropertyTrigger>() {
            @Override
            public void onResponse(AylaPropertyTrigger response) {
                //Newly Created Trigger is returned
            }
        },
        new ErrorListener() {
```

```
            @Override
            public void onErrorResponse(AylaError error) {
              //Log this error or Show a Toast message
            }
          });
```

In case there are already existing trigger fetch them as follows

```
aylaProperty.fetchTriggers(new Response.Listener<AylaPropertyTrigger[]>() {
            @Override
            public void onResponse(AylaPropertyTrigger[] response) {
              //Iterate through ArrayList and select the Trigger that needs an update
            }
        },
        new ErrorListener() {
            @Override
            public void onErrorResponse(AylaError error) {
           //Log this error or Show a Toast message
            }
        });
```

Once the AylaPropertyTrigger is fetched from Cloud do the following for creating email trigger App

```
 AylaPropertyTriggerApp triggerApp = new
AylaPropertyTriggerApp();
 triggerApp.setEmailAddress(emailContact.getEmail());
 triggerApp.configureAsEmail(emailContact, "[[property_name]]
[[property_value]]", null,
                     null);
 trigger.createApp(triggerApp,
            new Response.Listener<AylaPropertyTriggerApp>() {
           @Override
                        public void
onResponse(AylaPropertyTriggerApp response) {
                        //Newly Created Trigger App is returned
                        }
                    },
             new ErrorListener() {
                        @Override
                        public void onErrorResponse(AylaError
error) {
```

```
                                    //Log this error
                                });
```

The above case shows email Trigger App. In case of SMS or push message instead of configureAsEmail do the following

```
//In case of SMS
triggerApp.configureAsSMS(contact, message);
//In case of Push
triggerApp.configureAsPushAndroid(PushNotification.registrationId, message, null, null);
```