Rich Notification

Programming Guide

Version 1.1.3

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

Table of Contents

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

1. Overview

The goal of the Rich Notifications SDK is to offer a way for developers to reach users with a minimum effort. It provides the flexibility to personalize and brand content, making it compelling and actionable.

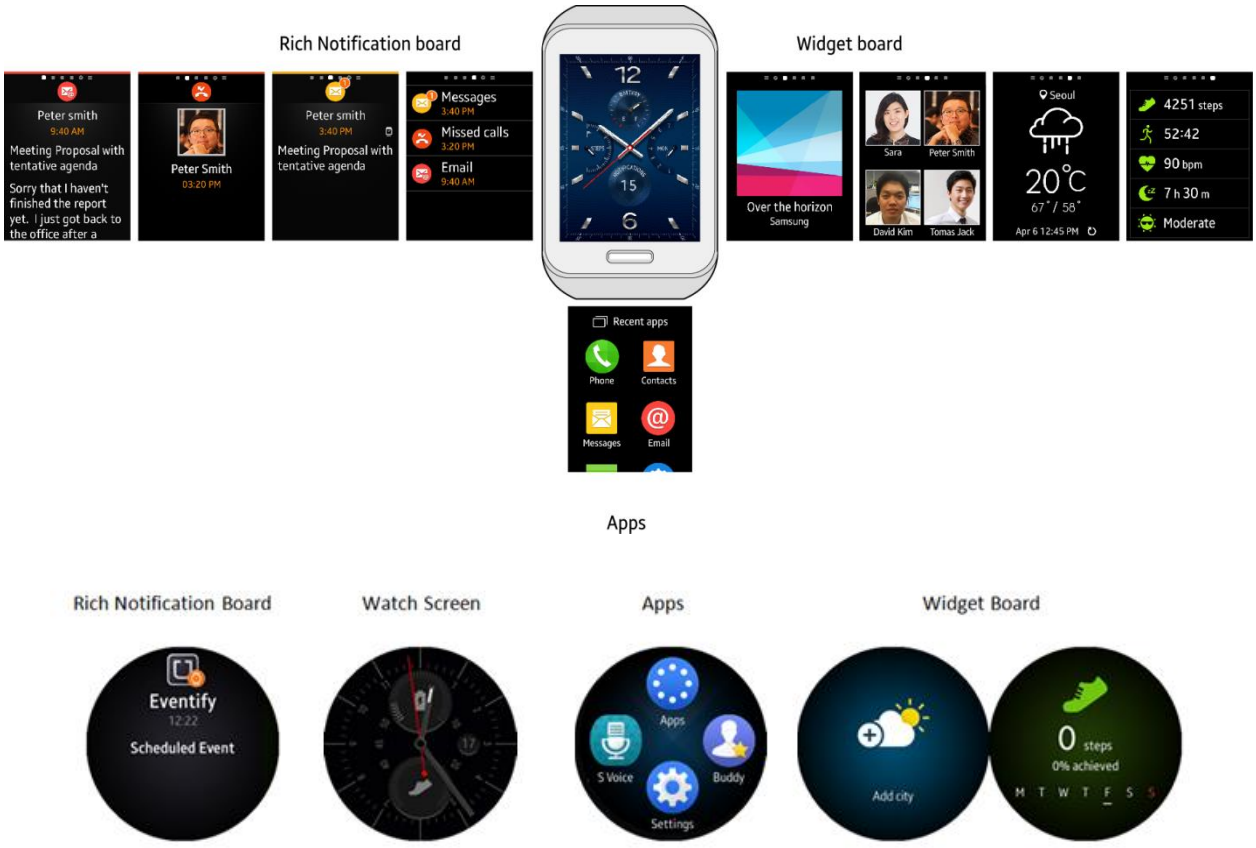


Figure 1 : Gear User Interface

1.1 Basic Knowledge

This chapter introduces the Rich Notification features on Gear .

1.1.1. Design Principles

Rich Notification has the following characteristics:

Glance-able : Rich Notification is a lightweight way for service providers to send updates to users in real-time. They are concise and informative, offering bite-sized updates that highlights the most important details at a glance.

Actionable : Actions are placed on the top level of Rich Notifications allowing users to take control with a single touch. Users can quickly perform actions such as responding to a message, navigating to a destination, or liking a photo without having to pull out their phone or dig through Apps.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

Delightable : Rich Notifications are designed with personalization in mind. They support full color graphics, rich text, and flexible content layouts. They offer a stylish and engaging way for users to interact with their favorite services.

1.1.2. Rich Notifications on Gear

There are four main components of the Gear User Interface: Home Screen, Rich Notifications, Widgets, and Apps. This document will focus on Rich Notifications.

1.1.3. Accessing Notifications

New Notifications : When a new notification arrives on the Gear, the screen is activated and a vibration is felt. The user can choose to interact with the notification or, after a period of inactivity, the screen will dim the notification andwill minimize it to the left side of the Home Screen.

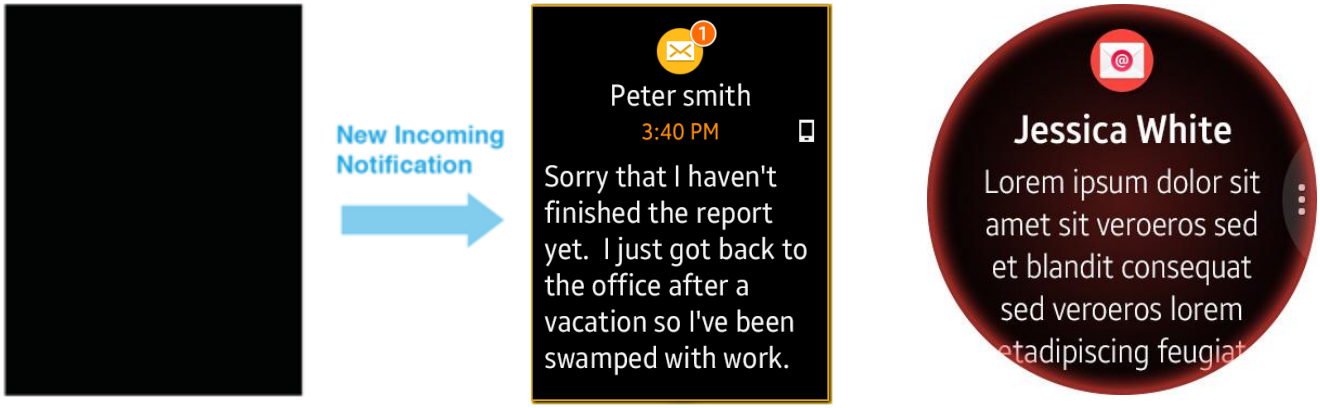
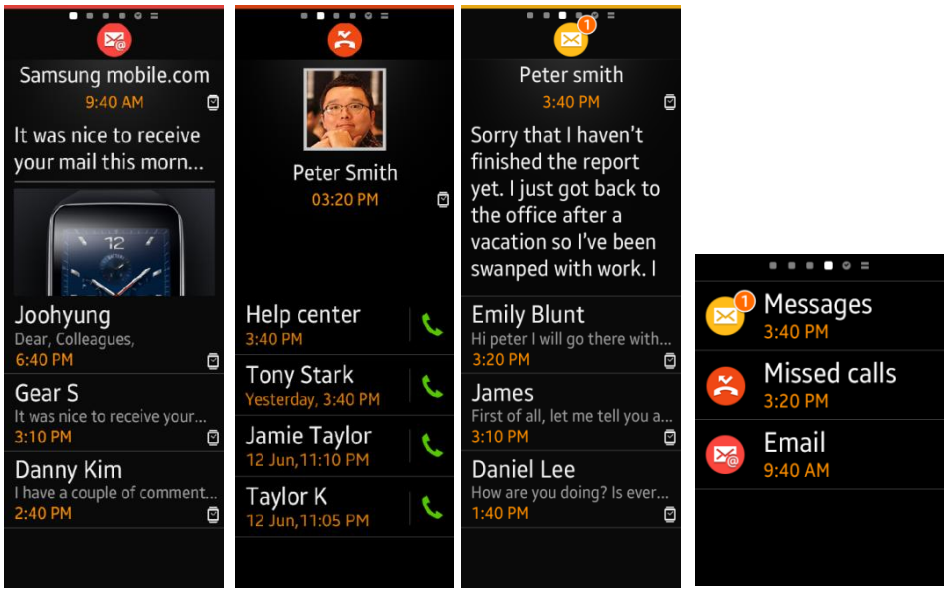


Figure 2 : New Notifications active the screeen.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

Previously Received Notifications : Users can return to previously received notifications by navigating to the left of the Home Screen.



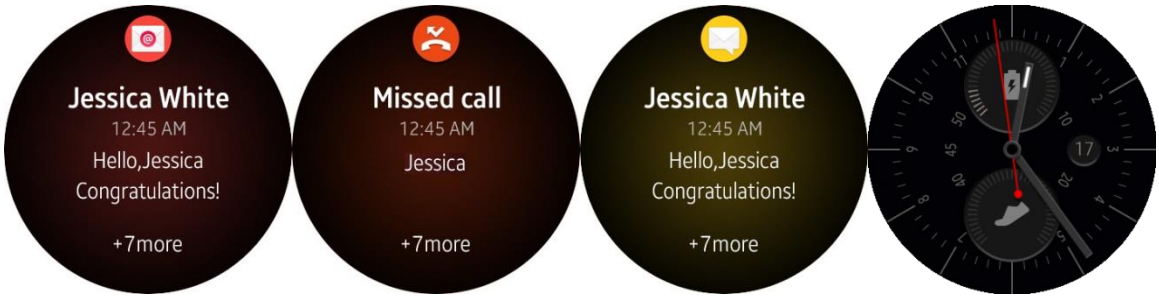


Figure 3 : Navigating to the left to view previously received notfications from different apps.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

1.1.4. Sorting Notifications

Notifications are automatically grouped by app and prioritized by arrival time.

Notifications are Grouped by App : Two or more notifications sent by the same app are automatically grouped together in a vertical *Panel*. Recent notifications are placed on top where they are most accessible.

Notification are Prioritized by Arrival Time : Two or more notifications sent by different apps are placed in the horizontal navigation with the most recent notification on the right.

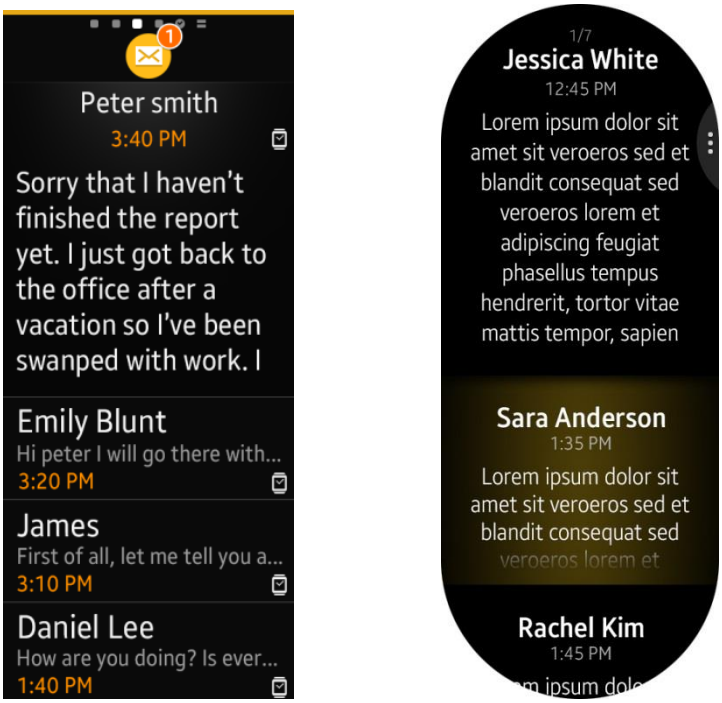


Figure 4 : Navigate down to view more notifications from an app.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

1.1.5. Content Structure

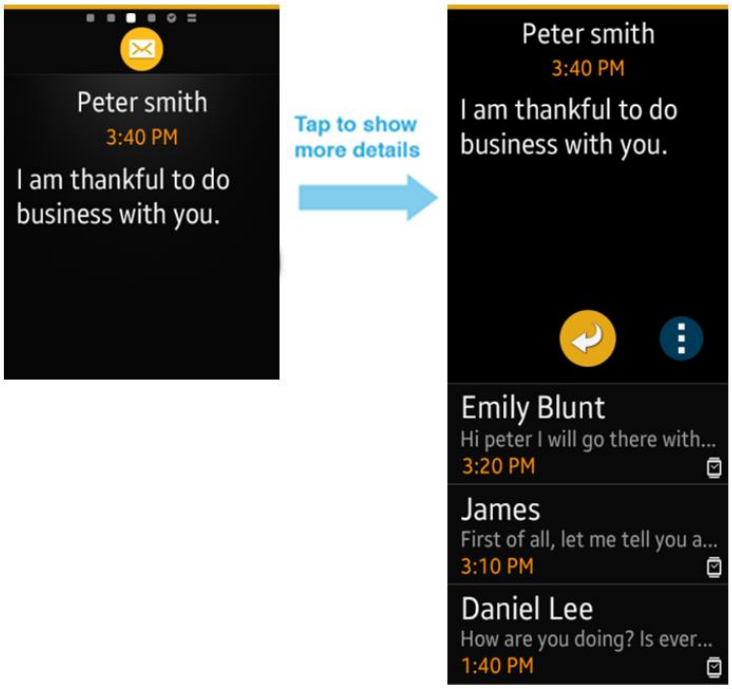
A standard notification is made of the following components:

Figure 5 : Rich Notification Components

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

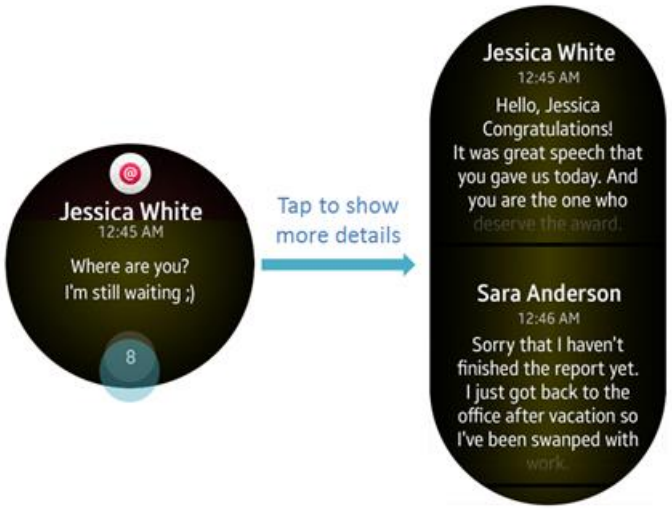
1.1.6. Tap to View More Details

Developers have the option to include secondary content with any notification. This content will be shown when the user taps on the body of the notification.



Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

In Gear S2, tap on notification count to expand all the notifications of the application.



Tap anywhere except notification count area to expand the latest notifcation of the application.

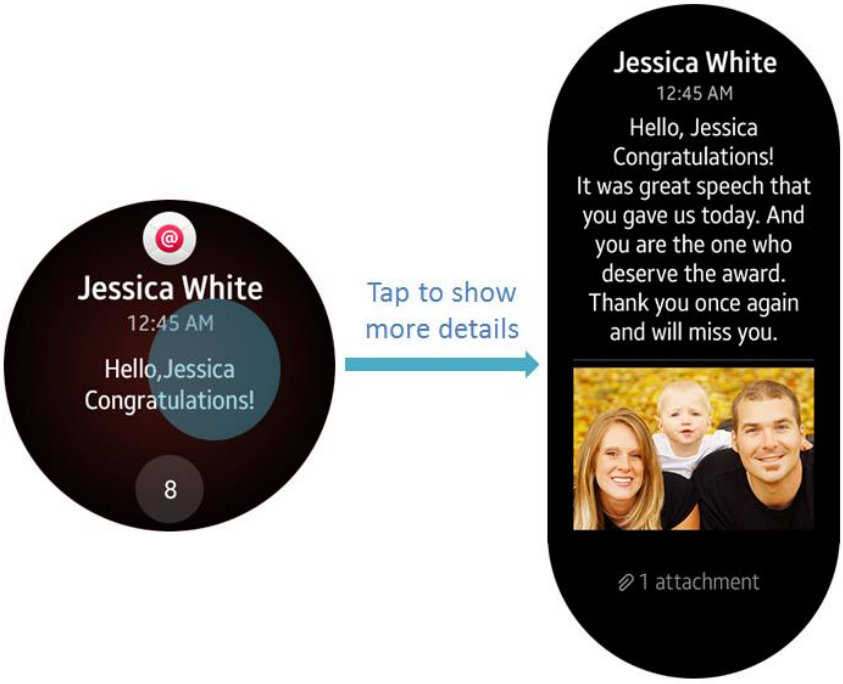


Figure 6 : Tap to expand a notification and view more details.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

1.1.7. Actions

Actions are an important part of Rich Notifications. They allow users to engage with the content and perform tasks. There are two types of Actions: Primary and More Options.

1.1.8. Primary Actions

Primary Actions are the most important action associated with the notification.

Figure 7 : Action task with a single tap.

1.1.9. More Options

The “More Options” icon produces a menu of additional actions that the user might want to take on the notification. These options are less accessible by design, placing the emphasis on the Primary Action.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

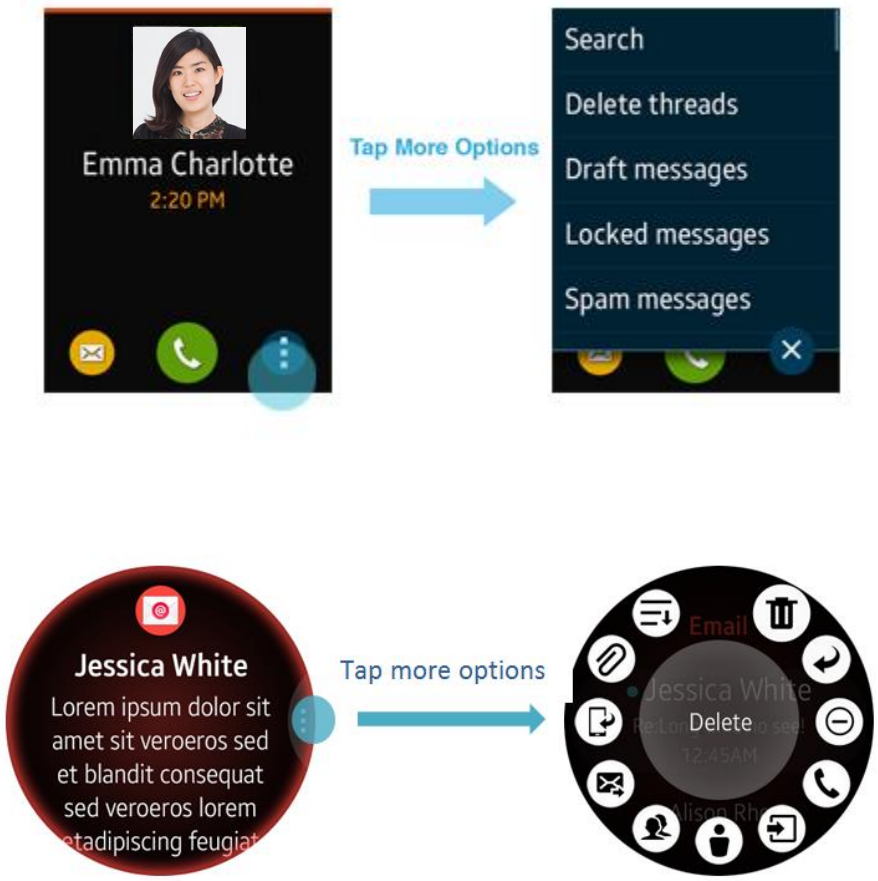
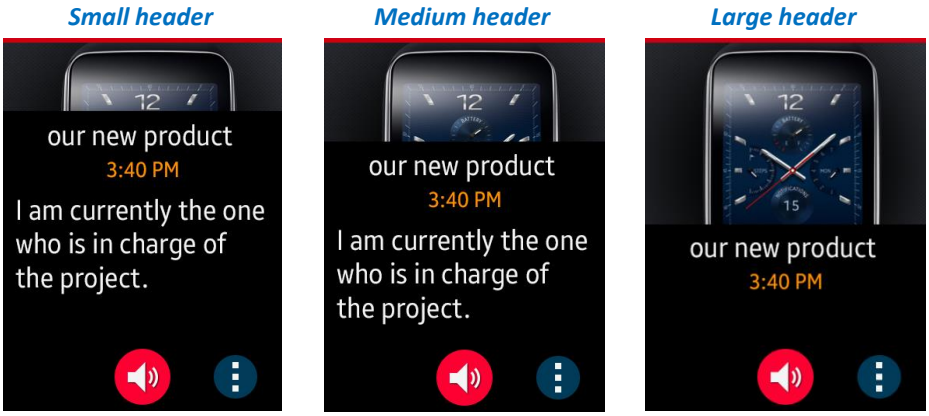


Figure 8 : Tap on the "More Options" icon for additional actions.

1.1.10. Templates

Developers can choose from a variety of templates that provide alternative visual layouts and content structures. There are two types of templates: Primary and Secondary.



Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

Figure 9 : Example Primary Templates.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

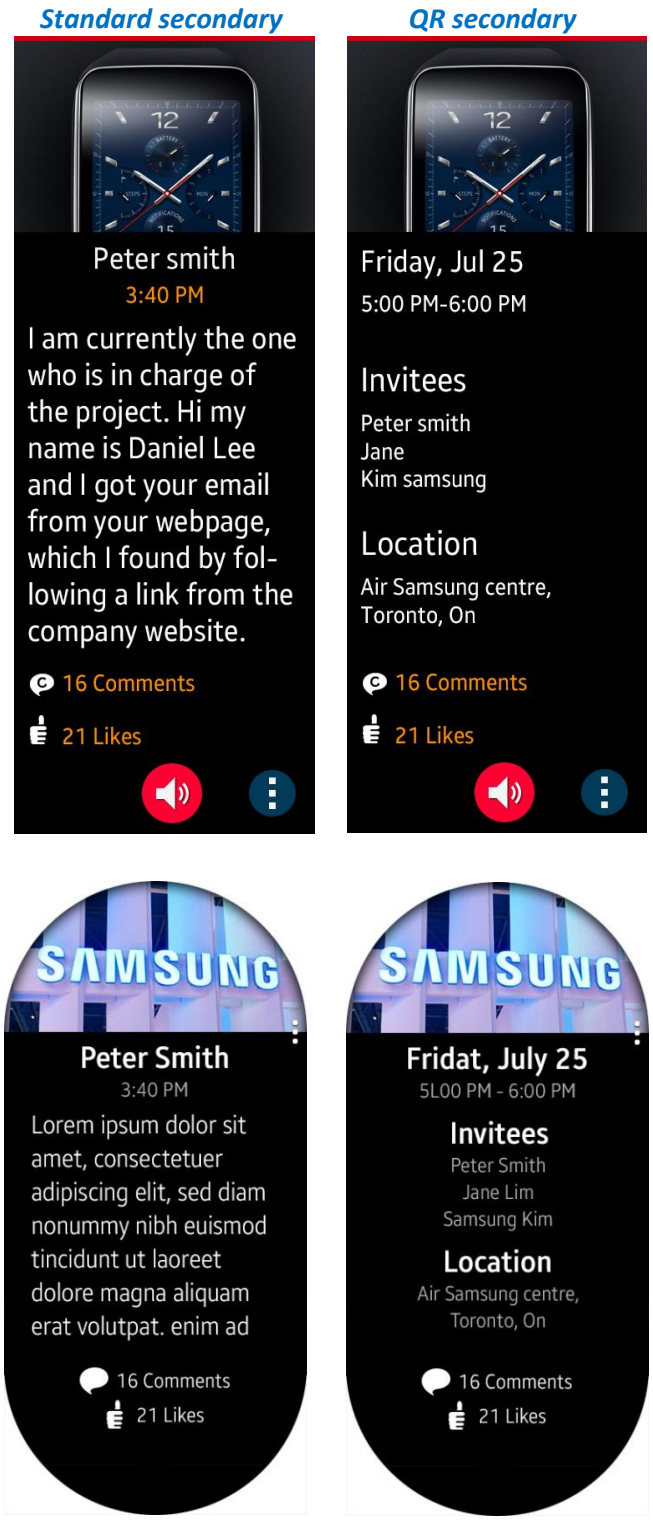


Figure 10 : Example Secondary Templates (Optional).

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

1.1.11. Using Templates and its fields

Primary and Secondary templates can be mixed and matched in a Notification. Most template fields are optional. Developers have the flexibility to choose which fields to populate and which to leave empty.

1.2 Architecture

The following figure shows the architecture of the Rich Notification:

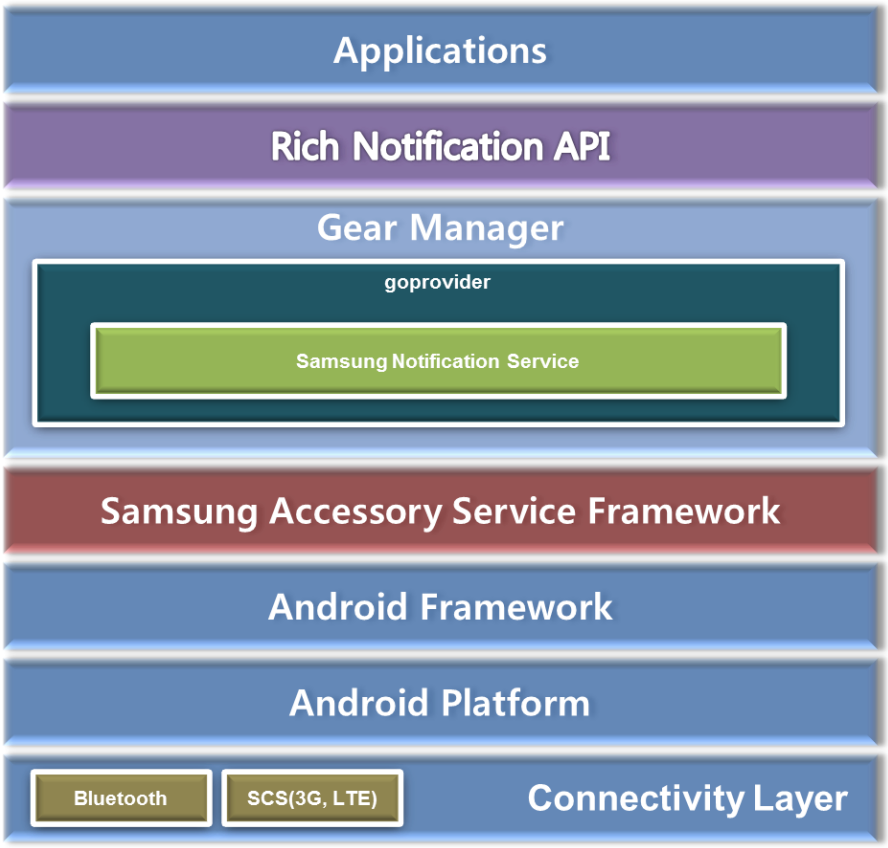


Figure 11 : Architecture

The architecture consists of:

- Applications: One or more applications that use Rich Notification.
- GearManager: Gear Manager Application.
- Goprovider : Service providers for the wearable.
- Samsung Notification Service : A service for sending the rich notification to the wearable.
- Samsung Accesory Service Framework : A framework for connecting devices.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

1.3 Class Diagram

The following figure shows the relationship between classes and interfaces of the `RichNotification`:

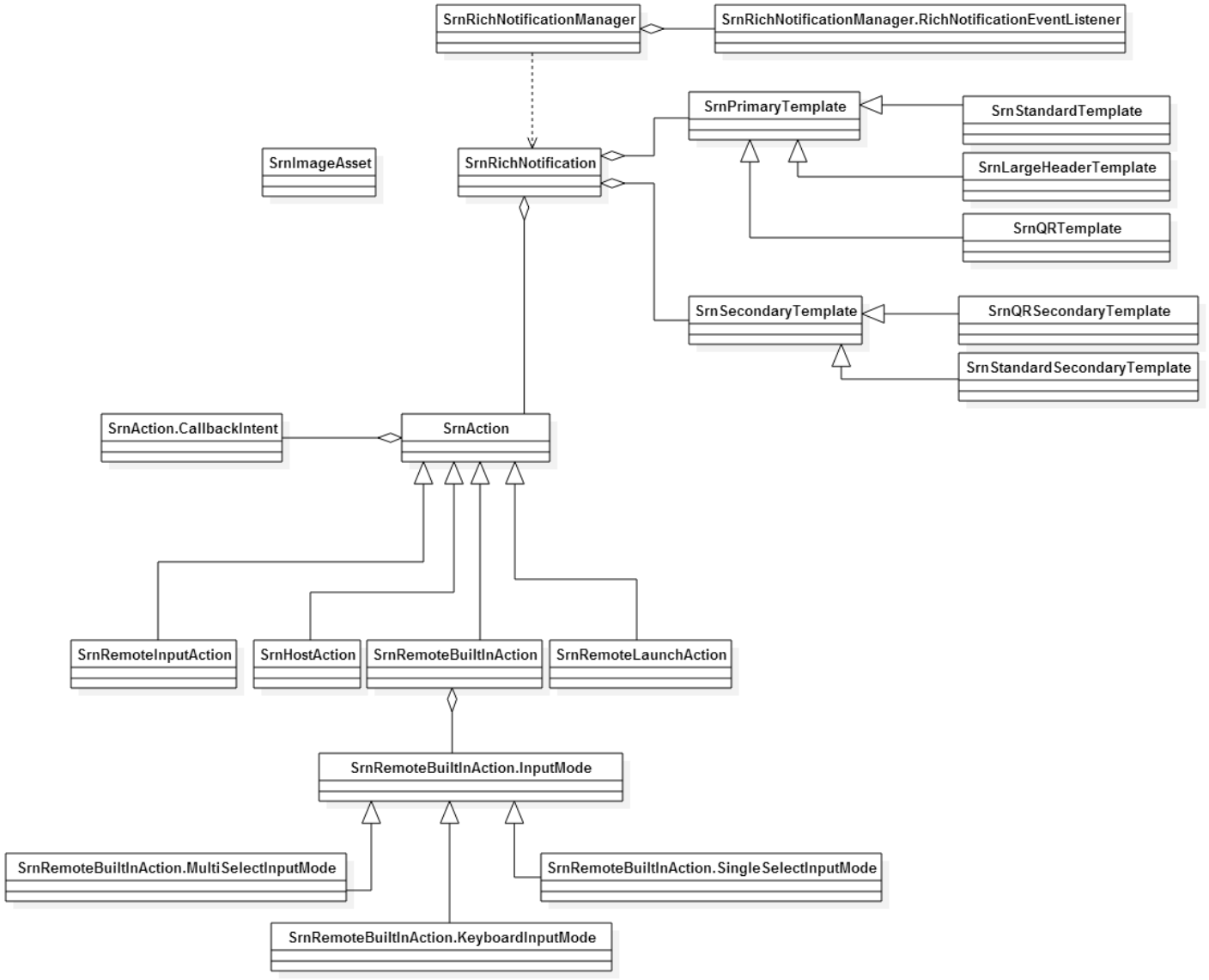


Figure 12 : Rich Notification Class diagram

Theinterfaces and classes are described in the following table.

Interface / Class	Description
SrnRichNotificationManager	Sends the notification to the wearable device.
SrnRichNotificationManager.EventListener	Listens for status of the notifications and the actions.
SrnRichNotification	Contains all the information of the notification.
SrnImageAsset	ImageWrapper for Rich Notification.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

SrnPrimaryTemplate	Base class of primary templates.
SrnSecondaryTemplate	Base class of secondary templates.
SrnStandardTemplate	Works best with content that has image and text.
SrnLargeHeaderTemplate	Works best when the image is a priority.
SrnQRTemplateTemplate	Works best with QR payment notifications.
SrnStandardSecondaryTemplate	Works best with paragraph text and optional image.
SrnQRSecondaryTemplate	Works best with list content and and an image.
SrnAction	Base class of actions which provides a mechanism that allows Rich Notification to do something on the host or wearable device.
SrnAction.CallbackIntent	Provides ways on how the intent will be launched on the host device.
SrnHostAction	The action will be handled by an Android application on the host device using Android intents.
SrnRemoteLaunchAction	The action will be handled by a Tizen application on the wearable device.
SrnRemoteBuiltInAction	Special built-in system actions that will be handled on the wearable device.
SrnRemoteInputAction	The action will prompt the user to provide additional input which will be handled by an Android application on the host device using an Android intent.
SrnRemoteInputAction.KeyboardInputMode	Allows a user to enter text using the system keyboard.
SrnRemoteInputAction.SingleSelectInputMode	Allows a user to choose a single choice from a list of options.
SrnRemoteInputAction.MultiSelectInputMode	Allows a user to choose zero or more choices from a list of options.

Table 1 : RichNotification Interfaces and classes

1.4 Supported Platform

Android KitKat 4.4.2 (API 19) or above

Note : It should be able to connect Gear with Gear Manager App.

1.5 Supported Features

The Rich Notification SDK provides the following features:

- Rich Notification : It provides the flexibility to personalize and brand content, making it compelling and actionable.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

- Action : Actions are an important part of Rich Notifications. They allow users to engage with the content and perform tasks. There are two types of Actions: Primary and More Options.
- Template : Developers can choose from a variety of templates that provide alternate visual layouts and content structures. There are two types of templates: Primary and Secondary
- Number of supported Notifications : In order to support synchronization and pending forward of notification, the total number of notifications, which could be delivered without synchronization or remove from host device, and connected wearable device is limited by 100 notifications.

1.6 Components

- Components
  - richnotification-v1.1.3.jar
  - sdk-v1.0.0.jar
  - gson-2.2.4.jar or above
- Package to be imported:
  - com.samsung.android.sdk.richnotification

1.7 Importing Libraries

To import Rich Notification libraries to the application project:

1. Add the following files richnotification-v1.1.3.jar, sdk- v1. 0.0.jar and gson-2. 2.4.jar(or later version of gson) file to the libs folder of your Android Application in Eclipse:
   - richnotification-v1.1.3.jar
     The client lib file of the richnotification package.
   - sdk-v1.0.0.jar
     The client lib file for the Samsung SDK. To use any SamsungSDK, this lib file must be included.
   - gson-2.2.4.jar or above

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

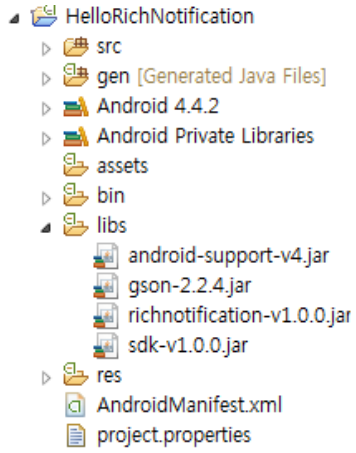


Figure 13 : libs folder in Eclipse

2. Add the following permissions to your Android manifest file:

   RichNotification needs the following permission. If these are not added in the `AndroidManifest.xml` file, the initialization will fail with `SecurityException`.

```
<uses-permission android:name="com.samsung.wmanager.ENABLE_NOTIFICATION"/>
<uses-permission android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY"/>
```

   If you don't add the permission,

   - For Samsung device,
     i. Android 4.4.2 (KitKat) and above: SecurityException is thrown and your application doesn't work.
     ii. Prior to Android 4.4.2 (KitKat): No exception. And the application works properly.
   - For other companies,
     i. No exception. and the application will work properly.

   If you use SrnRemoteBuiltInAction with "OperationType.CALL", it is required to add the following permission.

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

   If you use SrnRemoteBuiltInAction with "OperationType.SMS", it is required to add the following permission.

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

1.8 Application Registering Tip

When you publish your Rich Notification-enabled application to Google Play, add the **Gear_Rich_Notifications** keyword into the description field of the application registration page.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

By including this keyword in the app description, Gear users can then discover Rich Notification-enabled apps through promotional banners in the Gear App Store that directly link to search results for Gear_Rich_Notifications in Google Play. Users can also search **Gear_Rich_Notifications** direclty from Google Play.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

2. Hello RichNotification

`Hello RichNotification`is a sample code that introduces the Rich Notification API

```
public class HelloRichNotificationActivity extends Activity implements
        SrnRichNotificationManager.EventListener {

    private SrnRichNotificationManager mRichNotificationManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Creates an instance of Srn.
        Srn srn = new Srn();
        try {
            // Initializes an instance of Srn.
            srn.initialize(this);
        } catch (SsdkUnsupportedException e) {
            // Handles erros
        }

        mRichNotificationManager = new SrnRichNotificationManager(this);

        // Starts the manager.
         mRichNotificationManager.start();

        Bitmap bgBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.noti_bg);
        Bitmap myAppIconBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.ic_launcher);

        SrnRichNotification myRichNotification = new SrnRichNotification(this);

        SrnImageAsset myAppIcon = new SrnImageAsset(this, "app_icon", myAppIconBitmap);
        myRichNotification.setIcon(myAppIcon);

        myRichNotification.setTitle("Raptors vs. Nets");

        SrnStandardTemplate myPrimaryTemplate = new SrnStandardTemplate(HeaderSizeType.MEDIUM);

        SrnImageAsset myBackgroundImage = new SrnImageAsset(this, "noti_background", bgBitmap);
        myPrimaryTemplate.setBackgroundImage(myBackgroundImage);

        myPrimaryTemplate.setSubHeader("Today at 5 PM");
        myPrimaryTemplate.setBody("NBA finals pits Raptors against the Nets in game 6.");

        myRichNotification.setPrimaryTemplate(myPrimaryTemplate);

        SrnRemoteLaunchAction myAction = new SrnRemoteLaunchAction("Check In");

        Bitmap checkInIconBitmap = BitmapFactory.decodeResource(getResources(),
                R.drawable.ic_check_in);
        SrnImageAsset checkInIcon = new SrnImageAsset(this, "checkin_icon", checkInIconBitmap);

        myAction.setIcon(checkInIcon);
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

```
        myAction.setPackage("com.mypackage.myapp");
        myAction.setData(Uri.parse("checkin://venue_id=12345"));

        Intent resultIntent = new Intent(this, CallbackActivity.class);
        myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

        myRichNotification.addActionWithPermissionCheck(myAction);
        myRichNotification.setAlertType(AlertType.SOUND_AND_VIBRATION, PopupType.NORMAL);

        mRichNotificationManager.notify(myRichNotification);

         // Stops the manager.
         mRichNotificationManager.stop();

    }

    @Override
    protected void onResume() {
        super.onResume();

        mRichNotificationManager.registerRichNotificationListener(this);
    }

    @Override
    protected void onPause() {
        super.onPause();

        mRichNotificationManager.unregisterRichNotificationListener(this);
    }

    @Override
    public void onRemoved(UUID uuid) {
    }

    @Override
    public void onRead(UUID uuid) {
    }

    @Override
    public void onError(UUID uuid, ErrorType error) {
    }
}
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

3. Using the Srn Class

The Srn class provides the following methods:

- `initialize()` initializes Srn. You need to initialize the Srn package before you can use it. If the device does not support RichNotification, SsdkUnsupportedException is thrown.
- `getVersionCode()` gets the RichNotification version number as an integer.
- `getVersionName()` gets the RichNotification version name as a string.
- `isFeatureEnabled()` checks if a Srn package feature is available on the device.

```
Srn srn = new Srn();
try {
     // Initializes an instance of Srn.
     srn.initialize(this);
} catch (SsdkUnsupportedException e) {
     switch (e.getType()) {
         case SsdkUnsupportedException.DEVICE_NOT_SUPPORTED:
          // Indicates that device version is lower than Kitkat
          break;
     }
}

int versionCode = srn.getVersionCode();

String versionName = srn.getVersionName();
```

3.1 Using the initialize() method

The `Srn.initialize()` method:

- Initializes the Srn package.
- Checks if the device supports the Srn package.
- Checks if the Srnpackage libraries are installed on the device.

```
void initialize(Context context) throws SsdkUnsupportedException
```

If the Srn package fails to initialize, the `initialize()` method throws an SsdkUnsupportedException exception. To find out the reason for the exception, check the exception message.

3.2 Handling SsdkUnsupportedException

If an SsdkUnsupportedException exception is thrown, check the exception message type using SsdkUnsupportedException.getType().

The following types of exception messages are defined in the Srn class :

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

- DEVICE_NOT_SUPPORTED: The device does not support the Srn package.

3.3 Checking the Availability of RichNotificationPackage Features

You can check if anSrn package feature is supported on the device with the `isFeatureEnabled()` method. Pass the feature type as a parameter when calling the `isFeatureEnabled()` method. The method returns a Boolean value that indicates the support for the feature on the device.

Note **:** This is available in future releases.

```
boolean isFeatureEnabled(int type)
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

4. Using the RichNotification Package

This section describes how to use theRichNotification package in your application.

4.1. Creating the Rich Notification Manager

First, you need to create the **RichNotificationManager** that will be used to send the notification to the wearable device.

```
SrnRichNotificationManagerrichNotificationManager = new SrnRichNotificationManager(context);
```

4.2. Starting the Rich Notification Manager

After you create the RichNotificationManager, you must call start() to notify or dismiss the notification on the wearbale device.

```
richNotificationManager.start();
```

4.3. Stopping the Rich Notification Manager

When your application is terminated or the manager needs to be stopped, you must call stop() if the manager has started.

```
richNotificationManager.stop();
```

4.4. Creating a Rich Notification

Next, create a **RichNotification** which contains all of the information that makes up the notification itself (template, data to populate the template with, actions, etc.) We will dive more into the **RichNotification** itself a little later in this document.

```
SrnRichNotificationmyRichNotification = new SrnRichNotification(context);

SrnImageAssetmyAppIcon = new SrnImageAsset(this, "app_icon", myAppIconBitmap);
myRichNotification.setIcon(myAppIcon);

myRichNotification.setTitle("Raptors vs. Nets");
```

4.5. Checking Rich Notification Availability

If you want to check the connection of the wearable device which supportsRich Notification, use isConnected().

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

```
if(richNotificationManager.isConnected()) {
   ...
}
```

There is another way to be notified about Rich Notification availability using broadcast filtered by “com.samsung.wmanager.rich_notification.DEVICE_STATE_CHANGED”.

```
// AndroidManifest.xml
<receiver android:name=".MyBroadcastReceiver" >
    <intent-filter>
        <action android:name="com.samsung.wmanager.rich_notification.DEVICE_STATE_CHANGED" />
    </intent-filter>
</receiver>

// MyBroadcastReceiver.java
public void onReceive(Context context, Intent intent) {
    booleanisConnected = intent.getBooleanExtra("isConnected", false);

    if (isConnected) {
        ...
    }
}
```

4.6. Sending a Rich Notification

Once the rich notification is set up, use the **RichNotificationManager** to send it to the **RichNotificationService.** This call returns a UUID for the notification that you can store in your application to use in the future if you want to update or dismiss a previously sent rich notification.

Note **:** *Since Rich Notifications can persist on the wearable device, it is recommended that you store your notification’s UUID in a persistent data store on the host device to ensure it survives a host device reboot.*

```
UUID notificationId = richNotificationManager.notify(myRichNotification);
```

Note **:** If a Rich Notification from your application is not displayed in the device.In "Samsung Gear" of the mobile device, go to "Notifications" and make sure "Limit notifications" menu is unchecked. If this is checkedand the mobile device is being used,notifications will not be shown in the Gear.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

4.7. Updating an Existing Rich Notification

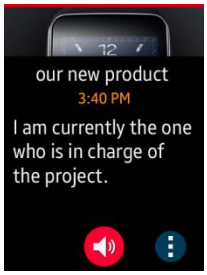
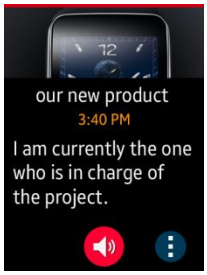
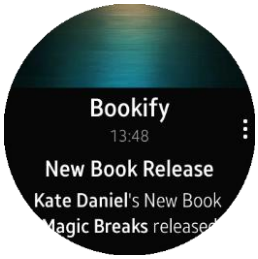
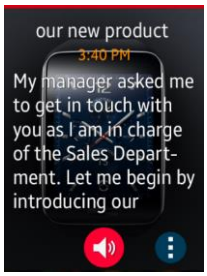
Using the UUID of a previously created Rich Notification allows you to update its content.

```
SrnRichNotificationmyUpdatedRichNotification = new SrnRichNotification(context, notificationId);
richNotificationManager.notify(myUpdatedRichNotification);
```

4.8. Using Templates

Multiple templates are provided to allow more control over how your Rich Notification will be displayed.

4.8.1. Specifying a Primary Template

Template Class Name	Description		
SrnStandardTemplate (Small header size)	Works best with content that contains a lot of text and little to no images.		
SrnStandardTemplate (Medium header size)	Works best with content that has image and text.		
SrnStandardTemplate (Full screen size)	Works best when the image is a priority and there is no text.		

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

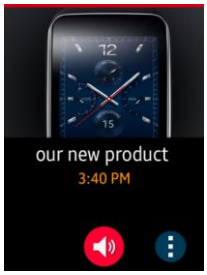
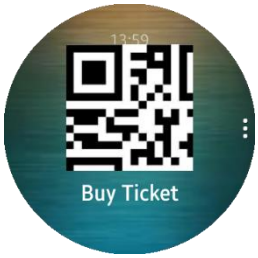
SrnLargeHeaderTemplate	Works best when the image is a priority.		
SrnQRTemplateTemplate	Works best with QR payment notifications.		

Table 2 : Primary Templates of RichNotification

The example of Primary Template codes are described below.

```
SrnStandardTemplatemyPrimaryTemplate = new SrnStandardTemplate(HeaderSizeType.MEDIUM);

SrnImageAssetmyBackgroundImage = new SrnImageAsset(context, "notification_background", imageBitmap);
myPrimaryTemplate.setBackgroundImage(myBackgroundImage);

SrnImageAssetmyAppIcon = new SrnImageAsset(context, "app_icon", myAppIconBitmap);
myPrimaryTemplate.setSubHeader("Today at 5 PM");
myPrimaryTemplate.setBody("NBA finals pits Raptors against the Nets in game 6.");
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

4.8.2. Specifying a Secondary Template

Secondary templates allow you to add more content to your Rich Notification. They are optional.

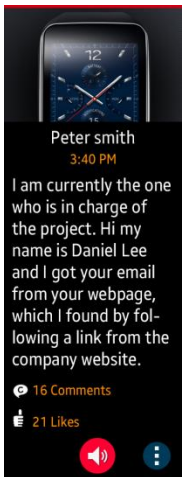
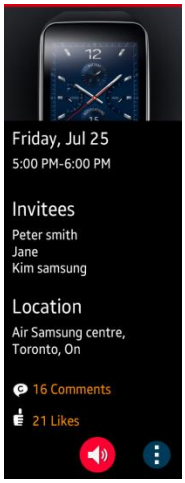
Template Class Name	Description		
SrnStandardSecondaryTemplate	Works best with paragraph text and optional image.		
SrnQRSecondaryTemplate	Works best with list content and and an image.		

Table 3 : Secondary Templates of RichNotification

The example of Secondary Template codes are described below.

```
SrnStandardSecondaryTemplatemySecondaryTemplate = new SrnStandardSecondaryTemplate();

SrnImageAssetmyContentImage = new SrnImageAsset(context, "content_image", contentImageBitmap);
mySecondaryTemplate.setImage(myContentImage);

SrnImageAssetmyLikeIcon = new SrnImageAsset(context, "like_icon", likeIconBitmap);
mySecondaryTemplate.setSmallIcon1(myLikeIcon, "21 likes");

SrnImageAssetmyCommentIcon = new SrnImageAsset(context, "comment_icon", commentIconBitmap);
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

```
mySecondaryTemplate.setSmallIcon2(myCommentIcon,"16 comments");

mySecondaryTemplate.setTitle("Pre-game analysis");
mySecondaryTemplate.setSubHeader("by Charles Barkley");
mySecondaryTemplate.setBody("The big question surrounding the Raps is whether ...");
```

Now that you have defined the primary and secondary templates and their associated content, you can add them to your Rich Notification.

```
SrnRichNotificationmyRichNotification = new SrnRichNotification(context);

myRichNotification.setIcon(myAppIcon);
myRichNotification.setTitle("Raptors vs. Nets");

myRichNotification.setReadout("NBA finals update", "Raptors take on the Nets ...");

myRichNotification.setPrimaryTemplate(myPrimaryTemplate);
myRichNotification.setSecondaryTemplate(mySecondaryTemplate);

richNotificationManager.notify(myRichNotification);
```

4.8.3. Text Formatting

Templates provide HTML tagsfor text styling. Styling texts can be usedto emphasis a part of a text or distinguish a text from a background image or color.

Tag	Description	Example
	Bold	This is bold
	Italic	I am italic
   	Hexadecimal color. The color is specified with an RGB value and optional alpha channel. The value always begins with a pound (#) character and then followed by the Red-Green-Blue-(Alpha) information.	Hello, Red

Table 4 : Tags for Styling Text

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

The example of styling text with supported tags is described below.

```
mySecondaryTemplate.setBody("<font color=\"#00F\"><i>The <b>big</b> question</i> surrounding the
Raps</font> is <b>whether ...</b>");
```

4.9. Actions

Note
addAction() and addActions() do not check permission for Gear side actions that may cause potential security problem. For that reason, we recommend using addActionWithPermissionCheck() and addActionsWithPermissionCheck() for added security.

Actions provide a mechanism that allows your Rich Notification to do something on the host or wearable device. You can specify one or more actions for your notification. There are currently four main action types supported:

Action	Description
SrnHostAction	The action will be handled by an Android application on the host device using Android intents.
SrnRemoteLaunchAction	The action will be handled by a Tizen application on the wearable device.
SrnRemoteBuiltInAction	Special built-in system actions that will be handled on the wearable device. Currently only "**CALL**" and "**SMS**" are supported.
SrnRemoteInputAction	The action will prompt the user to provide additional input which will be handled by an Android application on the host device using an Android intent. The input result can be retrieved from the Android intent's extras with the key: "**SAMSUNG_REMOTE_INPUT_RESULT**". The following input types are currently supported: "**KEYBOARD**", "**SINGLE_SELECT**", "**MULTI_SELECT**".

Table 5 : Actions of RichNotification

4.9.1. Create the List of Actions

Each Rich Notification can have one or more actions associated with it. The first action added to the list is considered the primary action, which is the action most likely to be taken by the user.

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

```
List<SrnAction>myActions = new ArrayList<SrnAction>();
```

4.9.2. Host Actions

Host actions will be handled by an Android application on the host device using Android intents.

```
SrnHostActionmyAction = new SrnHostAction("Check In");
SrnImageAssetcheckInIcon = new SrnImageAsset(context, "checkin_icon", checkInIconBitmap);

Intent resultIntent = new Intent(this, MyActivity.class);
...
myAction.setIcon(checkInIcon);
myAction.setToast("You were successfully checked in.");
myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

myActions.add(myAction);
```

4.9.3. Remote Launch Actions

Remote launch actions will be handled by an application on the wearable device. In this document, we assume that a Tizen device is connected. The example code of launching a specific applications is described below.

```
SrnRemoteLaunchActionmyAction = new SrnRemoteLaunchAction("Check In");

SrnImageAssetcheckInIcon = new SrnImageAsset(context, "checkin_icon", checkInIconBitmap);
myAction.setIcon(checkInIcon);
myAction.setPackage("com.mypackage.myapp");
myAction.setData(Uri.parse("checkin://venue_id=12345"));

// Optionally specifies an Android intent that will also be invoked.
Intent resultIntent = new Intent(this, MyActivity.class);
myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

myActions.add(myAction);
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

4.9.4. Remote Built In Actions

Remote built in actions are special system actions that will be handled on the wearable device. These actions use a default icon defined on the wearable device.

Supported Actions

Type	Description
CALL	Place a call on the wearable.
SMS	Send an SMS message from the wearable.

Table 6 : Remote Built in Actions of Gear

4.9.4.1. Call

```
SrnRemoteBuiltInActionmyAction = new SrnRemoteBuiltInAction("Call");
myAction.setType(OperationType.CALL);

// Optionally specifies the number to call.
myAction.setData(Uri.fromParts("tel", "+14157364480", null));

// Optionally specifies an Android intent that will be invoked.
Intent resultIntent = new Intent(this, MyActivity.class);
...
myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

myActions.add(myAction);
```

4.9.4.2. SMS

```
SrnRemoteBuiltInActionmyAction = new SrnRemoteBuiltInAction("SMS");
myAction.setType(OperationType.SMS);

// Optionally specifies the number to SMS.
myAction.setData(Uri.fromParts("sms", "+14157364480", null));

// Optionally specifies an Android intent that will be invoked.
Intent resultIntent = new Intent(this, MyActivity.class);
...
myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

myActions.add(myAction);
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

4.9.5. Remote Input Actions

Remote input actions will prompt the user to provide additional input which will be handled by an Android application on the host device using an Android intent.

Input Mode	Description
KEYBOARD	Allows a user to enter text using the system keyboard.
SINGLE_SELECT	Allows a user to choose a single choice from a list of options.
MULTIPLE_SELECT	Allows a user to choose zero or more choices from a list of options.

Table 7 : Remote Input Actions of RichNotification

4.9.5.1. Keyboard Input Mode

```
SrnRemoteInputActionmyAction = new SrnRemoteInputAction("Keyboard");

Intent resultIntent = new Intent(this, MyActivity.class);
...

myAction.setRequestedInputMode(InputModeFactory.createKeyboardInputMode()
    .setPrefillString("@name")
    .setCharacterLimit(140)
    .setKeyboardType(KeyboardType.NORMAL));

myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

myActions.add(myAction);
```

4.9.5.2. Select Input Mode

```
// Single Select
SrnRemoteInputActionmyAction = new SrnRemoteInputAction("Single Select");

Intent resultIntent = new Intent(this, MyActivity.class);
...

SingleSelectInputModesingleSelectInputMode = InputModeFactory.createSingleSelectInputMode();

//Choice with a label, value and icon.
singleSelectInputMode.addChoice("Black", "color_black", new SrnImageAsset(context, "black_icon",
blackImageBitmap));
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

```
//Choice with a label, value and icon.
singleSelectInputMode.addChoice("White", "color_white",  new SrnImageAsset(context, "white_icon",
whiteImageBitmap));

//Choice with a label and value.
singleSelectInputMode.addChoice("Red", "color_red");

myAction.setRequestedInputMode(singleSelectInputMode);
myAction.setDescription("Select your favoritecolor.");
myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

myActions.add(myAction);
```

```
// Multi Select
SrnRemoteInputActionmyAction = new SrnRemoteInputAction("Multi Select");

Intent resultIntent = new Intent(this, MyActivity.class);
...

MultiSelectInputModemultiSelectInputMode = InputModeFactory.createMutiSelectInputMode()
//Choice with a label, value and icon (selected).
   .addChoice("Black", "color_black", new SrnImageAsset(context, "black_icon", blackImageBitmap), true)

//Choice with a label, value and icon (selected).
    .addChoice("White", "color_white", new SrnImageAsset(context, "white_icon", whiteImageBitmap), true)

//Choice with a label and value.
    .addChoice("Red", "color_red")

//Choice with a label and value.
    .addChoice("Blue", "color_blue");


myAction.setRequestedInputMode(multiSelectInputMode);
myAction.setDescription("Select your favoritecolors.");


myAction.setCallbackIntent(CallbackIntent.getActivityCallback(resultIntent));

myActions.add(myAction);
```

Note : *Every choice item must have an icon to be displayed on the wearable device. If some of the items omit icons or set as null, all icons of all choice items will notbe displayed.*

Note : *A SelectInputMode must have a choice at the list.*

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

4.9.5.3. Handling Action Callbacks

You can set CallbackIntent to receive callbacks when actions are performed. There are three ways to be notified.

1) ActivityCallback will perform **context.startActivity()** on host device with **Intent.FLAG_ACTIVITY_NEW_TASK**.
2) BroadcastCallback will perform **context.sendBroadcast()**.
3) ServiceCallback will perform **context.startService()**.

You can retrieve the information like user type when the action is RemoteInputAction and it has KeyboardInputMode. The passed intent has a string extra named "extra_action_data".

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);

         String result = getIntent().getStringExtra("extra_action_data");

         if (result != null) {
              ...
         }
    }

}
```

In order to increasing security of your components, you need to add the permission in the manifest file:

```
<activity
    android:name=”MainActivity”
    android:permission=”com.samsung.accessory.permission.ACCESS_CALLBACK” />
```

4.10. Handling Rich Notification Callbacks

Let your activity or service implement the **SrnRichNotificationManager.EventListener** interface in order to get callbacks from the **RichNotificationService**. Once you have implemented the interface you need to register that class with the **SrnRichNotificationManager**.

Note **:** *Since Rich Notifications can persist on the wearable device, it is recommended that you store your notification's UUID in a persistent data storage on the host device to ensure it remains stored on the host device after reboot. Also, filtering the received UUIDs against stored ones greatly increases the security of your components.*

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

```
UUID notificationId = richNotificationManager.notify(myRichNotification);
```

```
MyActivity extends Activity implements SrnRichNotificationmManager.EventListener{
	public void onRemoved(UUID uuid) {
		// Called when a rich notification is removed.
	}

	public void onRead(UUID uuid) {
		// Called when a rich notification has been read.
	}

	public abstract void onError(UUID uuid, ErrorEnum error) {
		// Called when an error occurs. For example, if an action fails to be
		// taken on a remote device for some reason.
	}

	public void myRegisterFunction()
	{
	    ...
	    richNotificationManager.registerRichNotificationListener(myActivity);
	    ...
	}
}
```

4.11. Creating Existing Android Notifications on the Wearable Device

The **RichNotificationService** will automatically prevent Android Notifications from sending to connected peripherals. However, in some cases you might want to send the android notification to the connected wearables.In this case, you can use **setRouteCondition()** to send your Android notification to the wearable.

```
...
NotificationManagermNotificationManager =(NotificationManager)
	getSystemService(Context.NOTIFICATION_SERVICE);

NotificationCompat.BuildermBuilder =
	 new NotificationCompat.Builder(this)
	 .setSmallIcon(R.drawable.notification_icon)
	 .setContentTitle("My notification")
	 .setContentText("Hello World!");

Notification myAndroidNotification = mBuilder.build();

SrnRichNotificationManager.setRouteCondition(myAndroidNotification);
```

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

```
mNotificationManager.notify(mId, myAndroidNotification);
```

4.12. Making Relationship between Android Notification and Rich Notification

The **RichNotificationService**provides the way to make relationship between Android Notification and Rich Notification. If you want to treat the Android Notification and Rich Notification together in case of READ, DELETE and SYNC operation, you can use **setAssociatedAndroidNotification()** to make a relation between them.

```
...
NotificationManagernotificationManager =(NotificationManager)
       getSystemService(Context.NOTIFICATION_SERVICE);

NotificationCompat.BuildermBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");

Notification myAndroidNotification = mBuilder.build();

notificationManager.notify(mId, myAndroidNotification);

SrnRichNotificationmyRichNotification = new SrnRichNotification(context);

...

myRichNotification.setAssociatedAndroidNotification(mId);

richNotificationManager.notify(myRichNotification);
```

Note : *setRouteCondition() and setAssociation() could not be used at the same time for a single notification.*

Copyright © Samsung Electronics, Co., Ltd. All rights reserved.

Copyright

Copyright © 2014 Samsung Electronics Co. Ltd. All Rights Reserved.

Though every care has been taken to ensure the accuracy of this document, Samsung Electronics Co., Ltd. cannot accept responsibility for any errors or omissions or for any loss occurred to any person, whether legal or natural, from acting, or refraining from action, as a result of the information contained herein. Information in this document is subject to change at any time without obligation to notify any person of such changes.

Samsung Electronics Co. Ltd. may have patents or patent pending applications, trademarks copyrights or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give the recipient or reader any license to these patents, trademarks copyrights or other intellectual property rights.

No part of this document may be communicated, distributed, reproduced or transmitted in any form or by any means, electronic or mechanical or otherwise, for any purpose, without the prior written permission of Samsung Electronics Co. Ltd.

The document is subject to revision without further notice.

All brand names and product names mentioned in this document are trademarks or registered trademarks of their respective owners.

For more information, please visit http://developer.samsung.com/