1. 添加notification 过程就是通过notification\_stack\_scroller这个NotificationStackScrollLayout来addview ExpandableNotificationRow
2. ExpandableNotificationRow的创建：

createNotificationViews：

LayoutInflater inflater = (LayoutInflater) mContext.getSystemService(

Context.LAYOUT\_INFLATER\_SERVICE);

row = (ExpandableNotificationRow) inflater.inflate(R.layout.status\_bar\_notification\_row,

parent, false);

row.setExpansionLogger(this, entry.notification.getKey());

row.setGroupManager(mGroupManager);

row.setHeadsUpManager(mHeadsUpManager);

row.setRemoteInputController(mRemoteInputController);

row.setOnExpandClickListener(this);

protected void onFinishInflate() {

super.onFinishInflate();

mPublicLayout = (NotificationContentView) findViewById(R.id.expandedPublic);

mPublicLayout.setContainingNotification(this);

mPrivateLayout = (NotificationContentView) findViewById(R.id.expanded);

mPrivateLayout.setExpandClickListener(mExpandClickListener);

mPrivateLayout.setContainingNotification(this);

mPublicLayout.setExpandClickListener(mExpandClickListener);

}

**row.setExpansionLogger(this, entry.notification.getKey());**

当用户点击expand 按钮的时候最终在NMS中onNotificationExpansionChanged函数里打印notification\_expansion的event log

2016-11-17 10:27:27 的屏幕截图

通过notification中的remote view为NotificationContentView添加相应的click 事件和子view。

final StatusBarNotification sbn = entry.notification;

entry.cacheContentViews(mContext, null);

final RemoteViews contentView = entry.cachedContentView;

final RemoteViews bigContentView = entry.cachedBigContentView;

final RemoteViews headsUpContentView = entry.cachedHeadsUpContentView;

final RemoteViews publicContentView = entry.cachedPublicContentView;

// bind the click event to the content area

NotificationContentView contentContainer = row.getPrivateLayout();

NotificationContentView contentContainerPublic = row.getPublicLayout();

mNotificationClicker.register(row, sbn);

// set up the adaptive layout

View contentViewLocal = null;

View bigContentViewLocal = null;

View headsUpContentViewLocal = null;

View publicViewLocal = null;

try {

setIconTint(contentView, sbn);

contentView.setTimeoutEnabled(true);

contentViewLocal = contentView.apply(

sbn.getPackageContext(mContext),

contentContainer,

mOnClickHandler);

if (bigContentView != null) {

setIconTint(bigContentView, sbn);

bigContentView.setTimeoutEnabled(true);

bigContentViewLocal = bigContentView.apply(

sbn.getPackageContext(mContext),

contentContainer,

mOnClickHandler);

}

if (headsUpContentView != null) {

setIconTint(headsUpContentView, sbn);

headsUpContentView.setTimeoutEnabled(true);

headsUpContentViewLocal = headsUpContentView.apply(

sbn.getPackageContext(mContext),

contentContainer,

mOnClickHandler);

}

if (publicContentView != null) {

setIconTint(publicContentView, sbn);

publicContentView.setTimeoutEnabled(true);

publicViewLocal = publicContentView.apply(

sbn.getPackageContext(mContext),

contentContainerPublic, mOnClickHandler);

}

}

O:

public void addNotification(StatusBarNotification notification, RankingMap ranking)

protected NotificationData.Entry createNotificationViews(StatusBarNotification sbn

protected void inflateViews(Entry entry, ViewGroup parent)

private void updateNotification(Entry entry, PackageManager pmUser,

StatusBarNotification sbn, ExpandableNotificationRow row)

ExpandableNotificationRow::void updateNotification(NotificationData.Entry entry)

NotificationInflater::void inflateNotificationViews()

mPendingNotifications.put

AsyncInflationTask::doInBackground

NotificationInflater::InflationProgress createRemoteViews

NotificationInflater::CancellationSignal apply

NotificationInflater::boolean finishIfDone

public void onAsyncInflationFinished(Entry entry) mPendingNotifications.remove

private void addEntry(Entry shadeEntry)

protected void addNotificationViews(Entry entry) mNotificationData.add(entry);

protected void updateNotifications()

void updateNotificationShade()