# NoticationManagerService

## 如何设置静音模式

NotificationManager.java

在setNotificationPolicy（...）方法内部，通过AIDL方式获得了一个service，该service的具体实现是在NotificationManagerService.java中。由service调用setNotificationPolicy（...）

### NoticationManagerService. setNotificationPolicy

IBinder的实现类中，被调用的方法setNotificationPolicy（...）里面，可以看到代码mZenModeHelper.setNotificationPolicy(policy)，一步步深入。

### ZenModeHelper. setNotificationPolicy

public void setNotificationPolicy(Policy policy) {

if (policy == null || mConfig == null) return;

synchronized (mConfig) {

final ZenModeConfig newConfig = mConfig.copy();

newConfig.applyNotificationPolicy(policy);

setConfigLocked(newConfig, "setNotificationPolicy");

}

}

获得了一个ZenModeCogfig的对象，并调用applyNotificationPolicy方法。

### ZenModeConfig.applyNotificationPolicy(Policy policy)

enModeConfig.java

applyNotificationPolicy(Policy policy)，该方法主要是对Policy进行解析，将解析出来的值赋给各个变量。然后在回到ZenModeHelper.java。

### ZenModeHelper.setConfigLocked

ZenModeHelper.java

setConfigLocked(newConfig, "setNotificationPolicy")方法中先对配置文件进行空判断和有效判断。然后判断配置文件相对于原始是否有变得，如果有，调用dispatchOnConfigChanged()，该方法内部是一个回调。

private void dispatchOnConfigChanged() {

for (Callback callback : mCallbacks) {

callback.onConfigChanged();

}

}

所以我们再次回到NotificationManagerService.java

### NotificationManagerService.onConfigChanged

找到onConfigChanged()，里面调用了savePolicyFile()，该方法通过handle发送消息，处理该消息时调用handleSavePolicyFile()，该方法内部调用writePolicyXml()，该方法通过ZenModeHelper对象再继续调用writeXml(...)；于是再次回到ZenModeHelper.java

### ZenModeHelper. writeXml

找到writeXml(...)方法，里面遍历每一个ZenModeConfig对象，逐个调用ZenModeConfig对象的writeXml(...)方法。再次回到ZenModeConfig.java。

### ZenModeConfig.java

恭喜你，终于到头了。原来真正写数据的地方在这里。系统将之前解析出来的数据信息，也就是用户设定的各种勿扰的配置信息写在xml里面。好了，到此为止，另外如何使这些数据生效，请等待笔者的下一篇文章吧！

### REF

<https://blog.csdn.net/firedancer0089/article/details/78224249>

<https://blog.csdn.net/w690333243/article/details/77942222>

注明出处。

# TODO

clearNotificationEffects

enforceStatusBarService();

long identity = Binder.clearCallingIdentity();

try {

mNotificationDelegate.clearEffects();

} finally {

clearEffects原理NotificationManagerService extends SystemService

# Toast的显示

有这么一段diamante

public void show() {

if (mNextView == null) {

throw new RuntimeException("setView must have been called");

}

INotificationManager service = getService();

String pkg = mContext.getOpPackageName();

TN tn = mTN;

tn.mNextView = mNextView;

try {

service.enqueueToast(pkg, tn, mDuration);

} catch (RemoteException e) {

// Empty

}

}

## NMS.showNextToastLocked

fg

当ToastRecord被添加到mToastQueue中后，NMS就会通过showNextToastLocked方法来显示Toast，下面的代码很好理解，需要注意的是，Toast的显示是由ToastRecord的callback来完成的，这个callback实际上就是TN对象的远程Binder，通过callback来访问TN中的方法是需要跨进程来完成的，最终被调用的对象是TN中方法发起在Binder线程池中。

|  |
| --- |
| void showNextToastLocked() {  ToastRecord record = mToastQueue.get(0);  while (record != null) {  if (DBG) Slog.d(TAG, "Show pkg=" + record.pkg + " callback=" + record.callback);  try {  record.callback.show(record.token);  scheduleTimeoutLocked(record);  return;  } catch (RemoteException e) {  Slog.w(TAG, "Object died trying to show notification " + record.callback  + " in package " + record.pkg);  // remove it from the list and let the process die  int index = mToastQueue.indexOf(record);  if (index >= 0) {  mToastQueue.remove(index);  }  keepProcessAliveIfNeededLocked(record.pid);  if (mToastQueue.size() > 0) {  record = mToastQueue.get(0);  } else {  record = null;  }  }  }  } |

## ITransientNotification