Notification API 客户端开发

A.设置环境

要在symbian平台上开发一款支持通知的Qt应用程序，你需要下载Qt SDK1.1， 其自动安装通知API组件。通知客户端API实现为一个Qt 插件。安装Qt SDK后，通知API可以用QPluginLoader机制来加载

本章描述如何安装开发支持通知的Qt客户端应用所需要的安装的软件和文件。也介绍了如何构建和运行一个示例客户端应用，这样你可以快速的测试一下通知发送。

接下来你可以看看示例应用中那些类和信号用来获取通知ID和接收通知。

准备条件

进行下面流程前，假设下面的条件：

- 你有Nokia账号。 如果还没有请现在注册一个.
- 你已经注册了你的服务，获得了 服务ID和应用ID，如果还没有请现在注册。

流程

要进行下面三步：

- 第一步：下载 Qt SDK 1.1
- 第二步：导入通知例子工程，设置模拟器环境
- 第三步：在Qt模拟器环境下 编译运行通知示例

第一步： 下载 安装Qt SDK 1.1

下载Qt SDK 1.1,安装到你的电脑上。 通知API组件自动随之安装。

支持的开发平台

下表列出当前版本通知API支持的开发平台

平台	编译器	模拟器支持	支持的目标设备
Windows 32 and 64 bit	MinGW	Yes	支持的设备列表
Windows 32 and 64 bit	VS 2005	No	
Windows 32 and 64 bit	VS 2008	Yes	
Linux 32	gcc	Yes	No target devices, only for testing/demo purposes
Linux 64	gcc	Yes	No target devices, only for testing/demo purposes
Mac	gcc	Yes	No target devices, only for testing/demo purposes

第二步：导入通知例子工程，设置模拟器环境

打开Qt Creator。导入例子工程，选文件-->打开文件或工程。选择 <QtSDK>/Examples/notificationexample/notificationexample.pro 点击打开。选择Qt 模拟器->Simulator Qt for MinGW 4.4 然后结束。 编辑文件<QtSDK>/Simulator/OviNotifications/ovinotifications.ini 设置为你的账号凭证信息：

```
[username] string=username
[password] string=password
```

第三步：在Qt模拟器中编译，运行通知示例工程

要编译运行在Debug模式下，选择notificationexample Debug然后点击运行。如下面窗口所示。 工程编译后，下面的模拟器窗口弹出：

在模拟器上点 Register（注册）然后检查应用返回状态是online（在线）。 **注意**：Qt模拟器始终指向沙盒环境。不能被用于产品环境。 你可以用沙盒环境想你的应用发通知

- a.打开开发者控制台
- b.点My Services（我的服务）选项卡，在服务列表中选择 example.com。、
- c.在发送通知区域（见下图），设置使用nokia account id（Nokia账户ID）发送通知。
- d.在id栏输入你的Nokia Account ID ("your_username@ovi.com"), 在payload栏输入通知文本，点Push（推送）. 通知将显示在模拟器上。

example.com服务不支持 通知ID 发送通知。如果你有自己的服务，你可以在开发者控制台中使用通知ID给你设备上的示例应用发通知。 更多关于通知ID的用法，参见使用通知ID,要看示例通知应用如何发送通知ID，参看客户端示例应用。 **下一步** 现在你设置好了Qt的客户端通知API，你可以浏览一下示例应用代码中通知是如何实现的。

B.回顾示例应用

这个例子演示了如何使用Qt的通知API实现通知。客户端接收通知需要实现的主要步骤是：

- 设置应用ID和服务ID
- 载入插件连接信号

(c) Copyright Nokia 2013. All rights reserved.

- 注册应用
- 获取通知ID
- 接收通知消息

示例应用使用了通知API提供的一系列方法，参看API参考文档

向服务发送通知ID

示例包含了一个例子来发送通知ID给服务端。

定位示例应用工程

安装Qt SDK 1.1后，你可以在下面路径找到示例工程

```
/<QtSDK>/Examples/notificationexample/src/notificationexample.cpp
```

也可在这里下载。

设置应用ID和服务ID

要成功认证，你的客户端应用需要如下设置应用ID和服务ID：

```
const QString application_id = "com.example"; const QString service_id = "example.com";
```

上面的代码中“example.com”是公共服务"example.com"的服务ID，可以在开发者控制台使用这个ID来向示例发通知。注意使用这个公共服务时，你不能使用通知ID来指定通知的接收者，应为服务是公共的，能被任何人在开发者控制台来发送通知，因此通知ID不被支持。要使用公共的 example.com服务，你只能使用JID（用户名和应用ID）来指定通知的接收者。 要使用通知ID发送通知，注册你自己的服务，并替换示例中service_id 和 application_id的值。如下所示：

```
const QString service_id = "mynotifexample.com"; const QString application_id = "com. mynotifexample ";
```

载入插件连接信号

客户端通知API实现为一个Qt插件，使用QPluginLoader机制加载。下面的代码演示如何加载通知API，并连接所有提供的信号。

```
//Load the plugin using QPluginLoader
// ONE_PLUGIN_ABSOLUTE_PATH macro is added to the project when CONFIG+=ovinotifications is added in project's pro file.
QPluginLoader *loader = new QPluginLoader(ONE_PLUGIN_ABSOLUTE_PATH);
QObject *serviceObject = loader->instance();
if (serviceObject) {
     success = true;
     // Store the service interface for later usage
     iNotificationInterface = qobject_cast<OviNotificationInterface *>(
         serviceObject);
      // Connext signals to slots.
     connect(serviceObject,SIGNAL(stateChanged(QObject*)),
             this,SLOT(stateChanged(QObject*)));
     connect(serviceObject,SIGNAL(received(QObject*)),
             this,SLOT(received(QObject*)));
     connect(serviceObject,SIGNAL(notificationInformation(QObject*)),
             this,SLOT(notificationInfo(QObject*)));
     connect(serviceObject,SIGNAL(version(QString)),
             this,SLOT(versionResponse(QString)));
 }
```

注册应用

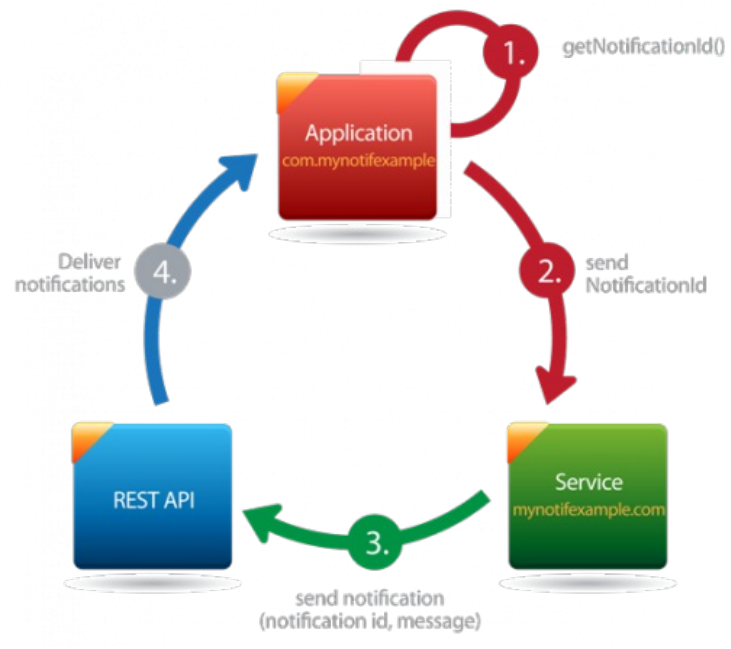
要向通知引擎注册应用，你需要使用registerApplication 方法，如下所示。但session状态改变时，发出OviNotificationSession::stateChanged信号， 这个信号返回一个OviNotificationState object，包含应用的状态。状态包括：online，offline，connecting（在线，离线，连接中） 更多关于应用状态的信息，参看客户端API概览。

```
iNotificationInterface->registerApplication(application_id);

// In the stateChanged slot

// State of the application has changed

OviNotificationState* state = static_cast<OviNotificationState*>(aState);

// Print out the session state on the screen switch (state->sessionState()) {
     case OviNotificationState::EStateOffline:
     {
         iTextConsole->append("Application is offline.");
         break;     }
     case OviNotificationState::EStateOnline:
     {
         iTextConsole->append("Application is online.");
         break;
     }
     case OviNotificationState::EStateConnecting:
     {
         iTextConsole->append("Application is connecting.");
         break;
     }
     default:
     {
         break;
     }
 }
```


(c) Copyright Nokia 2013. All rights reserved.

获取通知ID

推荐使用通知ID来指定接收者。通知ID由客户单申请，然后发送给服务端。如下所示：



注意：发送通知ID到服务端再通知服务以外进行。通知API并不提供发送通知ID给服务的机制。参看下面的一个例子演示了如何在应用中发送通知ID给你的服务。 要获取通知ID，首先连接 notificationInformation 信号如下：

```
// connecting to the notificationInformation signal
connect(serviceObject,SIGNAL(notificationInformation(QObject*)),
         this,SLOT(notificationInfo(QObject*)));
```

然后在session object上调用 getNotificationInformation方法如下：

```
iNotificationInterface->getNotificationInformation(service_id);
```

当通知ID获取后，发出notificationInformation信号，相应的槽连接在这个信号上，如下：

```
// In the notificationInfo slot
// Cast QObject to OviNotificationInfo to access the methods
OviNotificationInfo* info = static_cast<OviNotificationInfo*>(aData);
// Show the requested notification id on the screen in message box
QMessageBox messageBox;
messageBox.setText("Notification Id");
messageBox.setInformativeText(info->notificationId());
messageBox.exec();
delete info;
```

接收通知消息

要接收通知，你的应用需要连接received 信号（如上面解释的）。然后 OviNotificationMessage 和OviNotificationPayload用来读取通知：

```
// In the received slot
OviNotificationMessage* notification = static_cast<OviNotificationMessage*>(aNotification);
// Show the received notification in the screen
OviNotificationPayload*  payload =  static_cast<OviNotificationPayload*>(notification->payload());
     iTextConsole->append("Application " + notification->senderInformation()
                         + "of the service "+notification->from()+ " service sent: \n"
                         + "'" + payload->dataString() + "'");
```

向服务端发送通知ID

下面的代码演示了向服务端发送通知ID的一种方法。本例中使用 HTTP POST方法向开发者控制台发送通知ID：

```
QByteArray dataArray = QUrl::toPercentEncoding( iNotificationId );
QString post_url = "https://account.nnapi.ovi.com/cm/ext/addnid";
QString post_data = "nid=" + dataArray + "&sid=" + service_id;
QUrl qUrl(post_url);

if (iHttpManager == NULL)
     {
         iHttpManager = new QNetworkAccessManager(this);
         connect(iHttpManager, SIGNAL(sslErrors(QNetworkReply *, QList<QSslError>)),
                 this, SLOT(sslErrors(QNetworkReply *, QList<QSslError>)));
     }

iHttpManager->post(QNetworkRequest(qUrl), post_data.toLatin1());
```


(c) Copyright Nokia 2013. All rights reserved.

注意这只是提供了一个例子，你可以选择自己发送通知ID的方法。 **下一步?**

```
现在你看过了示例代码，接下来我们将其安装到手机上。
```

例子代码

Media:NotificationChat.zip

C.在设备上安装

本章描述如何在Symbian手机上安装客户端应用。包含下面几个方面：

- 在手机上安装测试用客户端
- 打包客户端应用，以便安装到终端用户的手机上

准备条件 下面流程假设：

- 你已注册了你的服务，获得了服务ID，应用ID。如果还没做，请现在注册。
- 你在你的应用中设置了账户凭证信息，参考示例代码。

在手机上安装测试用客户端

在目标手机上开发测试你的应用你需要：

- 第一步：安装通知支持包
- 第二步：为目标手机平台编译你的应用并安装到手机上
- 第三步：选择开发环境（沙盒环境或产品环境）

这是在手机上测试你的客户端应用的流程。打包应用，然后才能安装到终端客户的机器上。参看下一节。

第一步：安装通知支持包

客户端要与通知服务器交流，必须安装通知支持包。你需要安装一到两个文件，如下表所列。注意你不需要向你的用户提供这些文件。Smart Installer负责给你的用户安装这些文件。

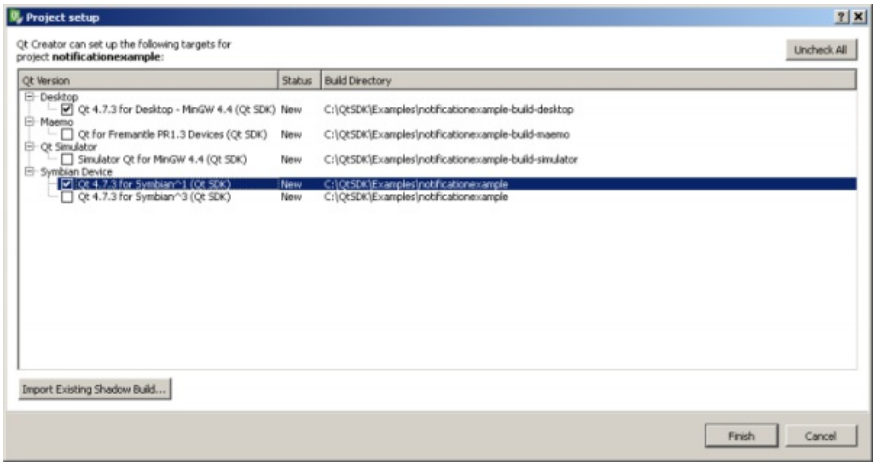
要安装的文件	路径
Notifications Support Package.sis（取决于手机型号）	/<QtSDK>/Symbian/sis/Symbian^1/OviNotifications/1.1.11106 或 /<QtSDK>/Symbian/sis/Symbian^3/OviNotifications/1.1.11106
qt_installer.sis (如果你的手机是S60 3rd Edition Feature Pack 2 或 5th Edition才需要安装)	/<QtSDK>/Symbian/sis/Symbian^1/Qt/<latest version>

第二步：编译应用

要编译你的应用：

```
打开Qt Creator, 选择开始-->程序-->Qt SDK-->Qt Creator。
```

导入你的工程，选择文件-->打开文件或工程。选择你的应用，打开，选 Qt 4.7.3 for Symbian^1 (Qt SDK) 或Qt 4.7.3 for Symbian^3 (Qt SDK)为你的目标设备。例如：



点结束。编译工程。生成<clientapplication>.sis。 例如，对示例应用会在notificationexample文件夹 生成notificationexample.sis ，使用PC套件安装这个文件。

第三步：选择开发环境

通知开发套件包含一个环境选择器应用。你必须安装这个应用，来启用沙盒环境。是如下表描述的sis文件：is available as a .sis file, as described in the table below:

设备	环境选择器 .sis文件	路径
S60 3rd Edition Feature Pack 2 或 5th Edition 手机	Notifications_Environment_Selector_S60_3rd_Ed_FP2_and_5th_Ed.sis	/<QtSDK>/Symbian/sis/Symbian^1/Notifications/1.1.11106
Symbian ^3	Notifications_Environment_Selector_Symbian3.sis	/<QtSDK>/Symbian/sis/Symbian^3/Notifications/1.1.11106

(c) Copyright Nokia 2013. All rights reserved.

开发测试中，你需要使用沙盒环境， 发布应用时，选择产品环境 默认情况下，客户端应用被配置成产品环境。 要选择沙盒环境： 打开环境选择器，激活沙盒环境

要选择产品环境： 只需再打开环境选择器，激活产品环境即可。 **下一步？** 现在你安装好了你的应用，你可以给他增加额外的功能，如，唤醒和通知界面。 参见 激活通知界面。

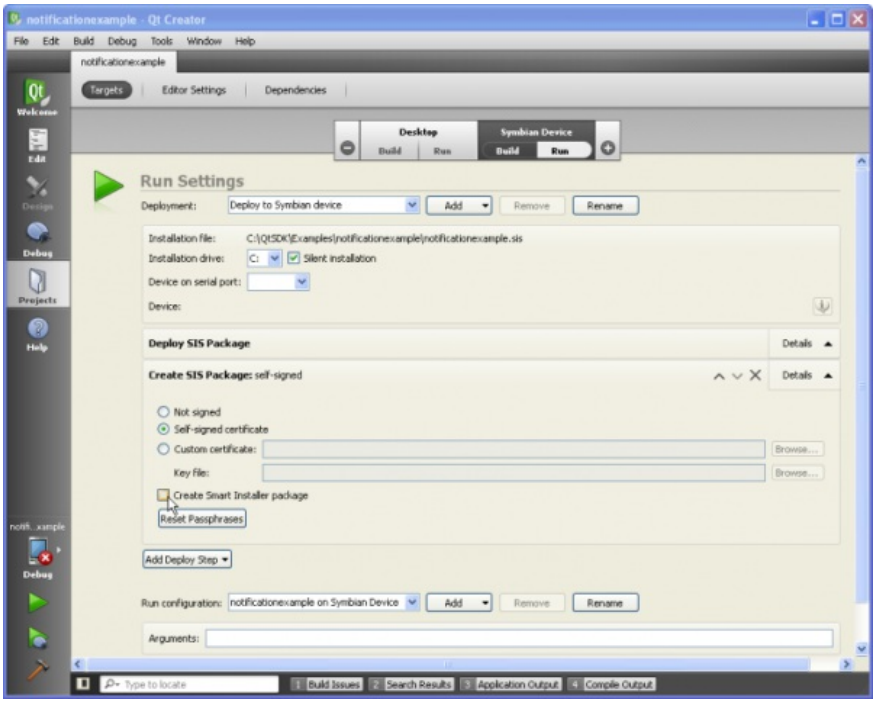
为终端用户打包客户端应用

安装客户端应用到终端用户的工作由Smart Installer处理. 要打包支持Smart Installer需要以下准备： 编辑你的应用 .pro文件（例如notificationexample.pro），加入这一行：

```
CONFIG+=ovinotifications
```

这行命令声明你的应用对通知支持包的依赖。这样当你的用户安装应用时，Smart Installer会检查你客户的手机是否包含相应的通知支持包。没有的话自动下载

创建并签名包含SmartInstaller的.sis文件，方法如下： 在Qt Creator，点工程选项卡， 在Symbian设备下点运行选项卡（Run）。 在运行设置窗口，展开Create Sis Package章节，下面的窗口显示：

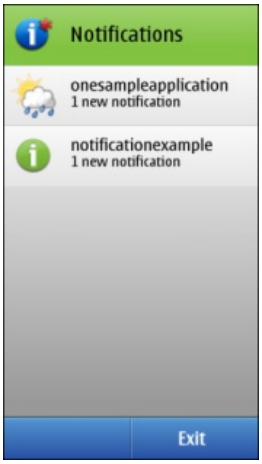


点创建Smart Installer包。 编译你的应用。 生成的<clientapplication>_installer.sis 文件包含你的应用所依赖的所有支持。终端用户将自动下载运行你应用所需要的文件，然后启动你的应用

D.通知界面

这个套件包含一个通知用户界面（UI），你可以使用这个界面来向你的用户显示通知，通知界面应用使用应用唤醒功能，可以在收到通知时唤醒没有启动的应用。 下图显示了一个通知界面应用的屏幕，列表列出了，收到消息的应用，点击上面的项，进一步查看通知。

(cc) Copyright Nokia 2013. All rights reserved.



通知界面演示视频

这段视频演示了如何使用通知界面：

激活通知界面

要激活通知界面应用，你的配置设置NotificationInterface，有两个重要的设置要设成true:

- setWakeUp
- setNotificationUi

如下例所示：

```
iNotificationInterface->setWakeUp(true);
iNotificationInterface->setNotificationUi(true, iIconData);
```

注意你可以设置一个显示在通知界面上的图标。 如上面的代码， "iIconData" 必须是base-64编码的图像数据。

设置交互 下表显示 setWakeUp 和setNotificationUi参数设置之间的交互：

	setNotificationUi=false	setNotificationUi=true
setWakeUp=false	客户端没启动，接收的通知丢弃	客户端没启动，接收的通知丢弃
setWakeUp=true	客户端启动，通知存储	通知UI弹出显示

setWakeUp 是主要的参数，当它关闭时，通知UI的状态是忽略，没有通知显示。

当唤醒功能打开但是通知UI关闭时：
==>当应用收到一个通知（同时服务端唤醒参数设为true），收到通知时客户端应用自动启动。

当唤醒功能打开但是通知UI打开时:
==>当应用收到一个通知（同时服务端唤醒参数设为true），通知显示在桌面（homescreen）， 用户可以选择显示这个通知。当只注册了一个应用时，应用直接打开。 当注册了多个应用时，通知UI应用显示，这时用户可选择查看某个通知应用。

注意通知只有在服务端和客户端的唤醒参数都设为true的时候才显示。参看服务端API参数以进一步了解服务端API唤醒参数设置。

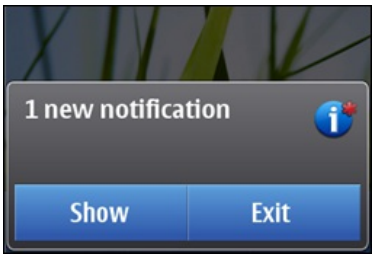
使用案例

单个应用注册并使用了通知UI应用。 在手机上启动示例应用（Notification Example）。 注册应用到服务。 选上Notification UI并允许唤醒（wake-up）。

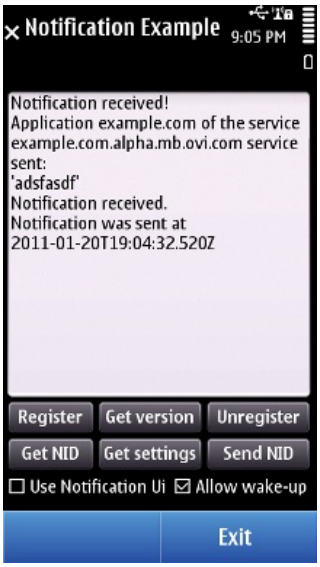
(c) Copyright Nokia 2013. All rights reserved.

不注销，退出应用，向应用发一条通知。你可以用沙盒环境下的控制台发一条通知。

如果你使用开发者控制台，注意发送前选上唤醒选项。 当通知UI功能激活时，通知显示在桌面上。

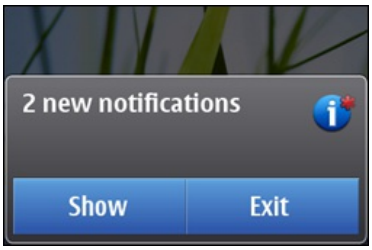


用户可以选择打开应用（show）。

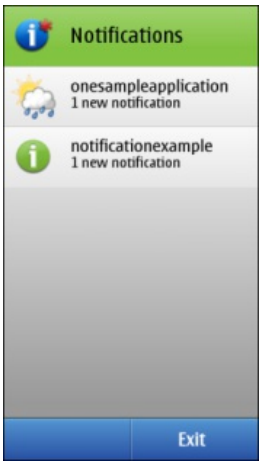


用户注册服务后，可以在通知UI应用中看到收到的通知。

多个应用注册，并启用通知UI应用。
当通知Ui功能处于激活状态，桌面会显示收到通知:



用户选择显示，通知UI应用出现：



通知Ui列出注册了的并收到通知的客户端应用，点选这些应用项，将打开相应的应用，显示通知。

为接收的通知设置声音提醒

如果你想当收到信息时，设备发出一个声音提醒，那么你可以通过Notifications API提供的有效设置来设置这项功能。如果想具有此功能，你的服务器端必须设置提醒标志，同时你的应用也需要设置声音提醒。

(c) Copyright Nokia 2013. All rights reserved.

当然，你要首先向Notifications API 注册你的应用。下面展示了客户端setAudioNotification API 和服务器端声音标识参数的交互:

	setAudioNotification=false	**setAudioNotification=true**
Audio flag on service side=false	没有声音提示	没有声音提示
Audio flag on service side=true	没有声音提示	有声音提示

存储和转交服务器功能

存储和转交是一个服务器端功能，此功能可以为用户保存通知消息，直到用户上线。
如果客户端应用处于离线，一个通知消息可以被保存以便过后发送。比如，当前没有为接收通知消息进行注册。当notification服务器端发现接收通知消息的客户端应用没有在线时，如果通知消息的"expires-at"字段的值被设置大于零，那么此通知消息将被保存。从服务器端来看，若客户端应用成功注销后，那么此客户端应用就处于离线状态了。如果处于离线状态的客户端应用在"expires-at"字段设置的有效期内，没有被设置成在线状态，那么保存的信息不会被发送，并且Notification 服务器端将自动删除此信息。
最多可以为每个用户/每个应用保存5条通知消息
最大保存时间为14天(由"expires-at"字段设置)
当通知消息被接收后，无论notification服务器端，还是应用客户端，都不会有什么返回的报告信息。如果服务器端需要此类报告信息，那么应用客户端需要报告给服务器端。
Notification API没有完成此功能。

```
Bug:
        目前此功能还存在问题，问题如下：
        若客户端应用处于离线(如，关闭手机的网络连接或关闭手机等)，在服务器端设置消息的"expires-at"字段值并发送通知消息，在一定的时间段内若你将客户端应用
        若在服务器端默认保持此网络连接这段时间内，将客户端应用处于离线，并在服务器端发送推送消息，那么服务器端会认为客户端应用收到了此推送消息，也就不会为
        服务器端默认保持此网络连接的时间大于20分钟小于一个小时。
        Notification API 开发团队正在解决这个问题，让我们期待此问题的解决吧。
```

参考链接

- Notification API 工程站点
- Notification API 白皮书
- Notification client API reference
- Qt 开发
- Qt Quick (Chinese)

(c) Copyright Nokia 2013. All rights reserved.

(c) Copyright Nokia 2013. All rights reserved.