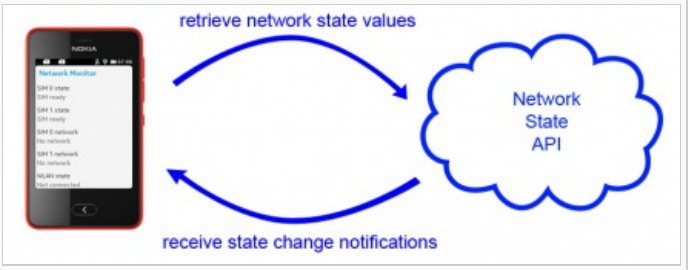
Network State API for Java ME

This article explains how to use the Network State API to monitor the network state on the Asha software platform.

30 Jun 2013

Introduction

The Asha software platform introduces the Network State API, that allows to retrieve the state of network, SIM and WLAN, and to receive notifications when any of those states change.



Network State API interaction modes illustrated

Network State API features

The Network State API is based upon three main classes:

- NetworkState gives access the network state of each of the installed SIMs
- SIMState gives access to the state of each of the installed SIMs
- WLANState allows to access the state of the device's WiFi Network

All those classes provide a common way to directly retrieve state values, or to be notified of state changes:

- a static `getState()` method immediately returns the desired state value
- the static `subscribeListener()` and `unsubscribeListener()` methods allow to subscribe to state value changes

The next sections show how those methods can be used in a sample app using the following LCDUI Form to display the values retrieved through the `Network State API`.

```
public class NetworkMonitorScreen extends Form
{
 public NetworkMonitorScreen()
 {
  super("Network Monitor");
 }

 // method used to add a log item to the Form
 private void addLog(String type, String message)
 {
  this.append(new StringItem(type, message));
 }
}
```

Network state

A Java app can directly retrieve the current network state for each of the installed SIM cards by using the NetworkState.getState method, passing as argument the `index` of the desired SIM. The method returns one of the possible network state values:

- NetworkState.NETWORK_STATE_HOME when SIM is in its home network
- NetworkState.NETWORK_STATE_NO_NETWORK when the SIM is not connected to any network
- NetworkState.NETWORK_STATE_ROAMING when the SIM is in a roaming network

The code snippet below returns a textual description for each of the possible states:

```
static String networkStateDescription(int state)
{
 switch(state)
 {
 case NetworkState.NETWORK_STATE_HOME:
  return "Home network";
 case NetworkState.NETWORK_STATE_NO_NETWORK:
  return "No network";
 case NetworkState.NETWORK_STATE_ROAMING:
  return "Roaming network";
 }
 return "Unknown state";
}
```

Using the method above, the sample `NetworkMonitorScreen` can show the current network state with the following code lines, requesting the state for the SIMs with `index` 0 and 1:

```
addLog("SIM 0 network", networkStateDescription(NetworkState.getState(0)));
addLog("SIM 1 network", networkStateDescription(NetworkState.getState(1)));
```

Network state of two SIM cards installed on a Nokia Asha 501

Subscribing to network state changes

A Java app can be notified of changes of the SIMs' network state by implementing the NetworkStateListener interface and subscribing to changes via the NetworkState.subscribeListener static method.

Implementing the `NetworkStateListener` interface requires the implementation of its networkStateChanged method, that accepts two arguments:

- the index of the SIM whose network state is changed
- the new network state value

The following code snippet shows how the sample `NetworkMonitorScreen` can implement the `NetworkStateListener` interface with the following implementation of the `networkStateChanged` method:

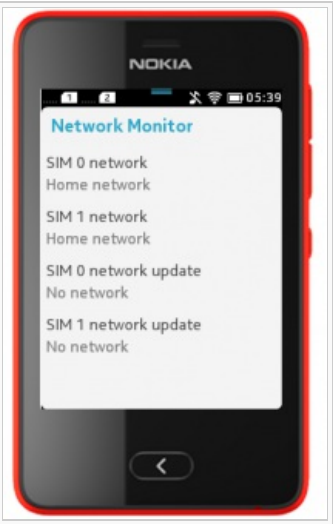
```
public void networkStateChanged(int simIndex, int state)
{
 String message = networkStateDescription(state);

 addLog("SIM " + simIndex + " network update", message);
```

```
}
```

Finally, to properly receive notifications, the `NetworkMonitorScreen` must subscribe to state changes:

```
NetworkState.subscribeListener(this);
```

The following screenshot shows how the `NetworkStateListener` is notified when both SIM cards lose connection to network:



The Java app is notified when the SIMs are no more connected to network

Note: Network state changes can be simulated on the Asha emulator by using the *Tools -> Generate Event -> No Signal* menu option

SIM state

The state of any of the SIM installed on a Asha device can be retrieve by using the SIMState.getState method, passing as argument the index of the desired SIM. The method returns one of the possible state values for a SIM:

- SIMState.SIM_STATE_NO_SIM indicates a SIM not currently inserted in the device
- SIMState.SIM_STATE_NOT_READY indicates a SIM inserted but not still ready to be used
- SIMState.SIM_STATE_READY indicates a SIM ready to be used

Similarly to what done before, a utility method that converts those values to human-readable textual descriptions can be defined as follows:

```
static String simStateDescription(int state)
{
 switch(state)
 {
 case SIMState.SIM_STATE_NO_SIM:
  return "No SIM";
 case SIMState.SIM_STATE_NOT_READY:
  return "SIM not ready";
 case SIMState.SIM_STATE_READY:
  return "SIM ready";
 }
 return "Unknown state";
}
```

With the method above, the `NetworkMonitorScreen` can show the retrieved state of the two installed SIMs as shown below:

```
addLog("SIM 0 state", simStateDescription(SIMState.getState(0)));
addLog("SIM 1 state", simStateDescription(SIMState.getState(1)));
```

SIMs state retrieved on a Nokia Asha 501 with only one SIM inserted

Subscribing to SIM state changes

SIM state changes are notified to an instance of the SIMStateListener interface, specifically to its SIMStateChanged method. This method accepts two arguments:

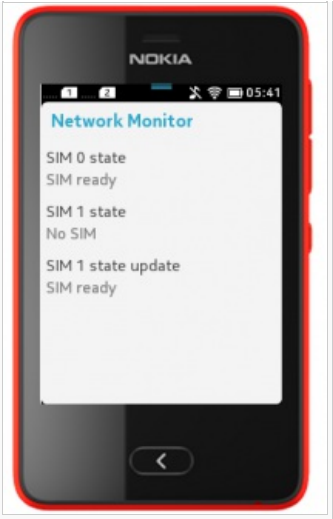
- the `index` of the SIM whose state changed
- the new state value

The following code snippet shows how the `NetworkMonitorScreen` can implement the `SIMStateChanged` to show on screen the new SIM state:

```
public void SIMStateChanged(int simIndex, int state)
{
 String message = simStateDescription(state);

 addLog("SIM " + simIndex + " state update", message);
}
```

Last thing, the `SIMStateListener` instance must subscribe to state changes via the `SIMState.subscribeListener` method:

```
SIMState.subscribeListener(this);
```



SIMStateListener called when inserting a second SIM

Note: A SIM hot-swap can be simulated on the Asha emulator by using the *Tools -> Generate Event -> SIM HotSwap* menu option

WLAN state

The `Network State API` also allows to monitor and retrieve the state of the device WLAN connection. The retrieval of WLAN state can be performed through the WLANState.getState method, that returns one of the following two values:

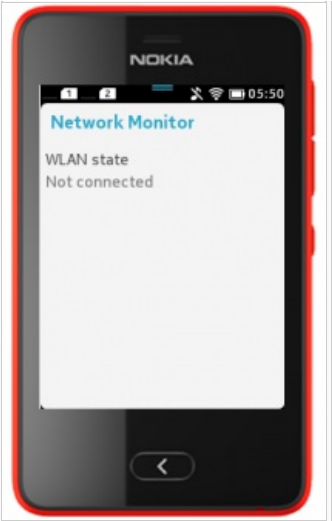
- WLANState.WLAN_STATE_CONNECTED if WLAN is connected
- WLANState.WLAN_STATE_NOT_CONNECTED if WLAN is not connected

Those states can be easily converted to textual descriptions with the following utility method:

```
static String wlanStateDescription(int state)
{
 switch(state)
 {
 case WLANState.WLAN_STATE_CONNECTED:
  return "Connected";
 case WLANState.WLAN_STATE_NOT_CONNECTED:
  return "Not connected";
 }
 return "Unknown state";
}
```

The method above can then be used to show the current WLAN state in the sample `NetworkMonitorScreen`:

```
addLog("WLAN state", wlanStateDescription(WLANState.getState()));
```



WLAN state retrieved on a Nokia Asha 501 not connected to a WLAN network

Subscribing to WLAN state changes

The WLANStateListener interface is responsible for receiving notifications of WLAN state changes, and defines a WLANStateChanged method whose implementation must handle those changes.

The following sample implementation uses the `NetworkMonitorScreen.addLog` method to show the new WLAN state value on screen:

```
public void WLANStateChanged(int state)
{
 String message = wlanStateDescription(state);

 addLog("WLAN state update", message);
}
```

To actually receive notifications, the listener must subscribe to changes via the WLANState.subscribeListener static method.

Connecting on a WLAN network properly calls the WLANStateChanged method

Use cases

The `Network State API` is useful in all those scenarios where having knowledge of the device network connection can improve an app's user experience, including:

- checking the network and WLAN state before performing a network connection
- checking if WLAN is available before downloading large chunks of data from network

- warning the user if the device SIM is on a roaming network
- checking the SIM and network state before attempting to perform a phone call or sending a SMS/MMS message

Source code

Full source code of sample Java app illustrated in this article can be downloaded here: Media:WikiNetworkExample.zip

Summary

This articles illustrates the features offered by the *Network State API*, showing how each of them can be used to retrieve network state values and to receive notifications when those values change.

Further information is available on the Nokia Developer's Library and on the Network State API JavaDocs pages.