RCS-e stack

Wiki

Edition 1.1
Date: 18/08/2011

Author:
OrangeLabs, ASC Devices
Jean-Marc AUFFRET, jeanmarc.auffret@orange-ftgroup.com

Summary

1. How to build the RCS core stack? ... 3
2. How to build the RI application? ... 3
3. IMS authentication ... 5
4. SIPS ... 6
5. SIP NAT traversal ... 7
6. RTP NAT traversal ... 8
7. How to generate your IMS profile? ... 8
8. Automatic NAT detection ... 9
9. Trace management ... 9
10. Display name ... 10
11. Software release number ... 11
12. How to remove the RCS service ... 11
13. How to deactivate the RCS service ... 12
14. RCS permission ... 13
15. Dynamic RCS extensions ... 13
16. Session refresh management ... 14
17. AIDL remote exception ... 14
18. Media players ... 14
19. RCS API ... 15
20. Social presence sharing ... 15
21. Phone number format ... 15

Figures

Figure 1: Eclipse project for RCS core stack ... 3
Figure 2: Eclipse project for RCS RI application ... 5
Figure 3: IMS authentication configuration ... 6
Figure 4: SIPS configuration ... 7
Figure 5: SIP keep-alive configuration ... 8
Figure 6: Generate your IMS profile ... 9
Figure 7: Trace management ... 10
Figure 8: Display name configuration ... 11
Figure 9: RCS account management ... 12
Figure 10: RCS service management ... 13
Figure 11: Presence service ... 15
Figure 12: SIP contact format ... 16
Figure 13: Country code ... 17

1. How to build the RCS core stack?

Via Eclipse, create a new Android project from the existing source code located in the directory `\trunk\core`:

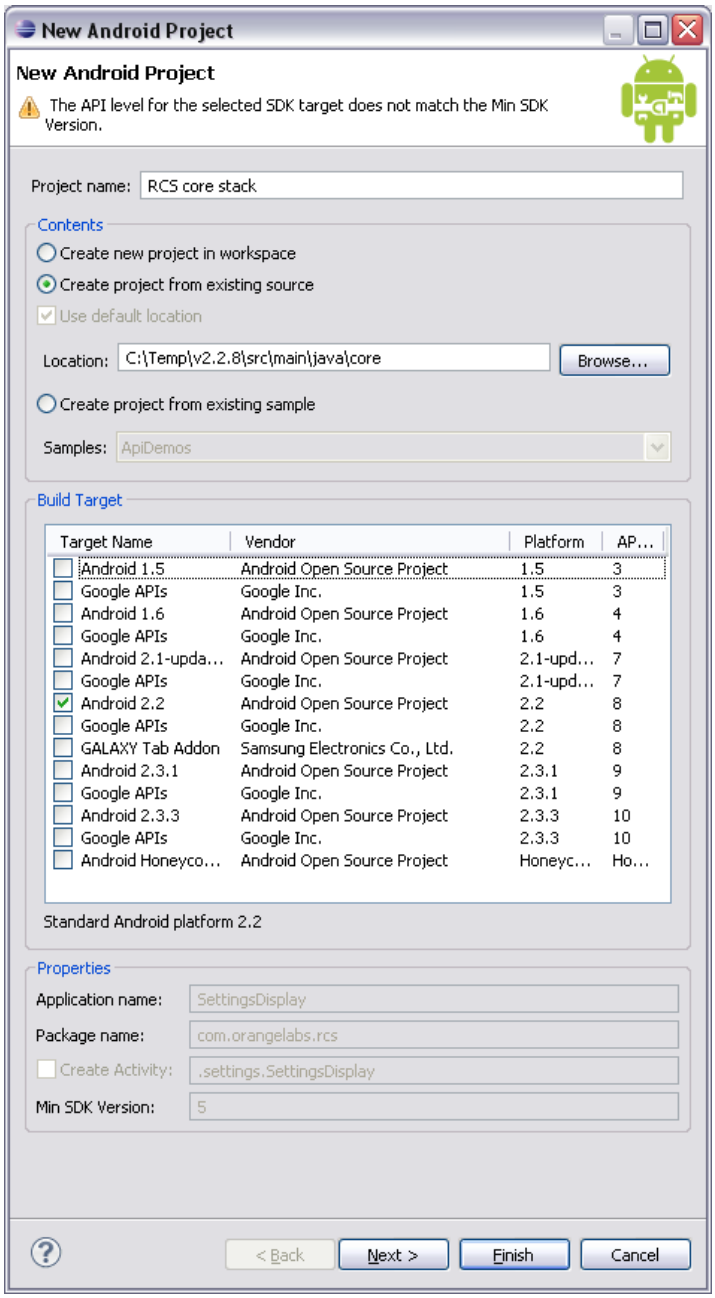


Figure 1: Eclipse project for RCS core stack

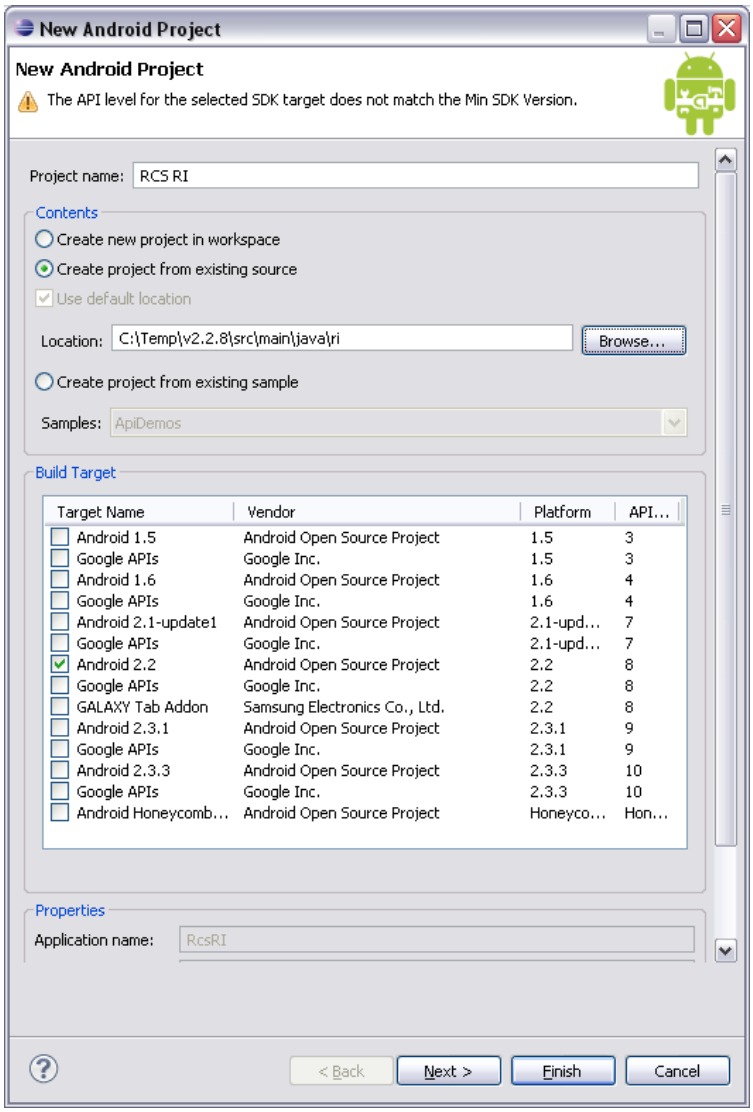
Via the Android framework, see the makefile file Android.mk.

2. How to build the RI application?

1. Create the RCS API jar file from the RCS core stack project by using the build file `\trunk\core\genapi.bat` or `\trunk\core\genapi.sh`. This build file extracts

only the required class files from the stack in order to generate the JAR file containing the RCS API.

2. Via Eclipse, create a new Android project from the existing source code located in the directory `\trunk\ri`:



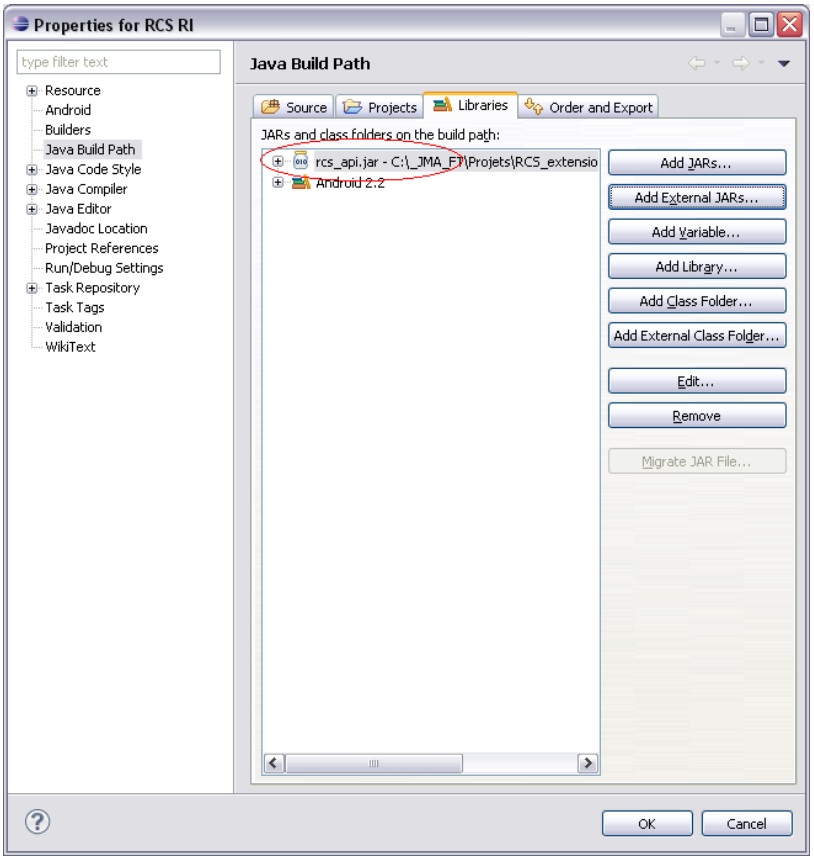


Figure 2: Eclipse project for RCS RI application

Note: do the same for any application using the RCS API (e.g. Chat application, widget).

3. IMS authentication

From a Mobile access, the RCS-e stack uses GIBA (early-IMS) by default to register to the IMS platform. It's possible to change the default authentication procedure (GIBA or DIGEST) via the provisioning application or via OTA. The GIBA procedure uses IMSI from SIM card to authenticate the end user: no need of login/password.

From a Wi-fi access, the RCS-e stack uses HTTP Digest by default to register to the IMS platform. The Wi-fi access supports only the HTTP Digest procedure.

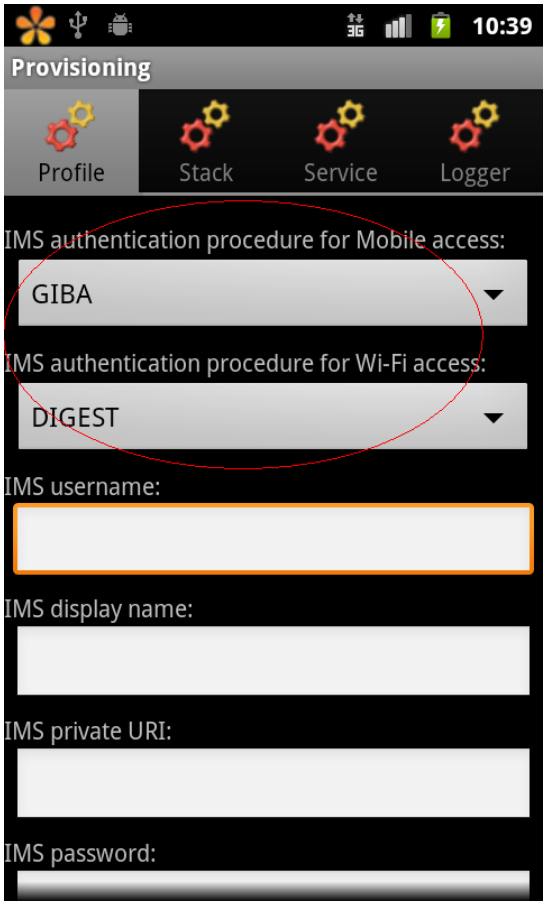


Figure 3: IMS authentication configuration

For HTTP Digest, the login/password is generated via the provisioning application or via OTA:

4. SIPS

SIPS only works with Android 2.3 or later.

The NIST SIP stack supports SIP over TLS. Normally, the SIPS should be activated behind a Wi-Fi hotspot in order to secure the SIP transport layer.

For Mobile or Wi-Fi access, the default connection type may be changed via the provisioning application. TLS must be set for SIPS connection.

TLS protocol uses one or two certificates to validate the connection. These certificates can also be set via the provisioning application. Certificates are selected from the "*.crt" files at the root of SDCARD. The "*.crt" files should have the following format:

```
-----BEGIN CERTIFICATE-----
MIIC2jCCAkOgAwIBAgIBATANBgkqhkiG9w0BAQUFADBtMQswCQYDVQQG
...
-----END CERTIFICATE-----
```

If the IMS uses a self-signed certificate, you should directly use this certificate in the RCS application. If the IMS uses a normal certificate, you should use the intermediate and root certificates.

Note: the P-CSCF of the IMS may need a particular port like 5061 (can be set via the provisioning application in Profile part).

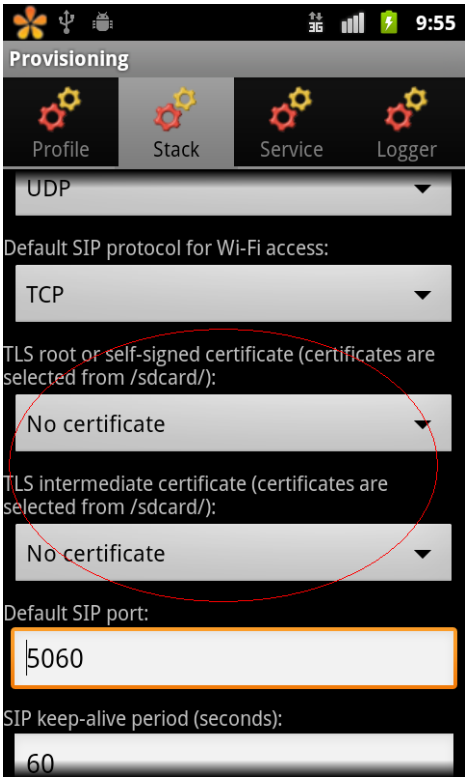


Figure 4: SIPS configuration

5. SIP NAT traversal

The NIST SIP stack supports SIP keep-alive procedure. When the RCS-e stack detects a NAT, the keep-alive procedure is automatically activated: a double CRLF is sent periodically in order to keep the NAT binding for SIP.

The keep-alive period is 60 seconds by default and may be changed via the provisioning application or via OTA. The keep-alive procedure may also be completely deactivated.

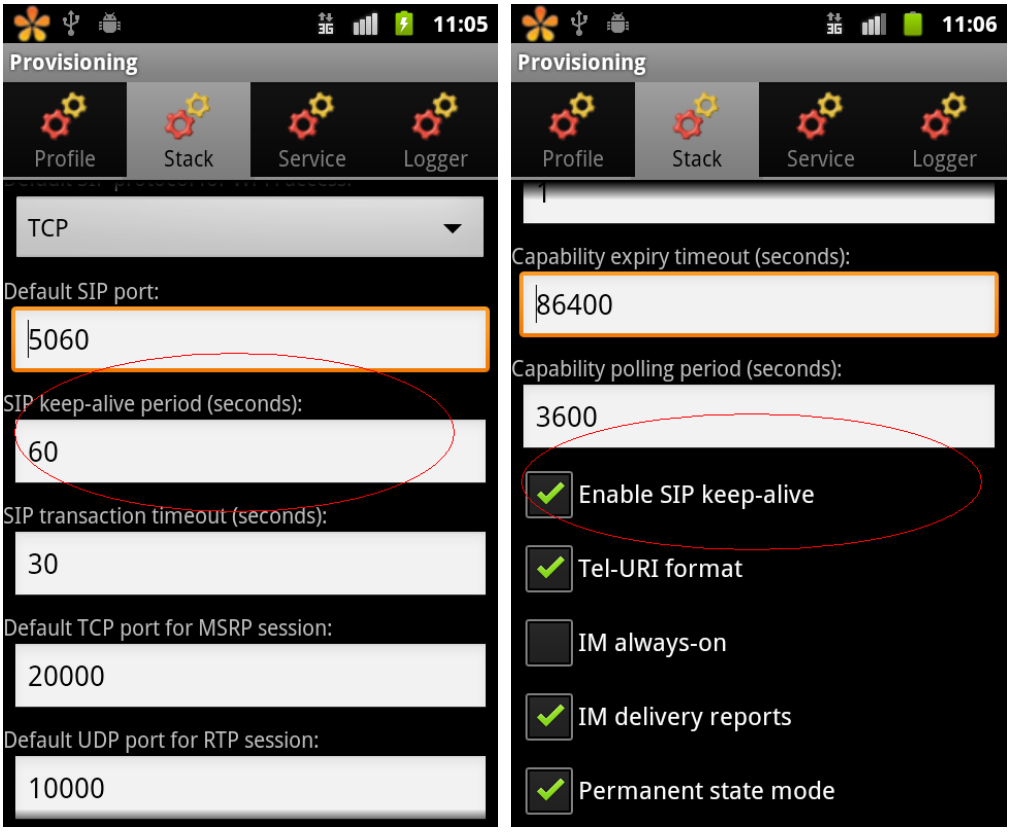


Figure 5: SIP keep-alive configuration

6. RTP NAT traversal

The RCS-e stack supports symmetric RTP stream (same used to send and to receive RTP data). A dummy packet with payload 12 is sent on the inactive leg (in case of unidirectional stream).

7. How to generate your IMS profile?

By default the following IMS profile is set in the RCS settings of the stack:

- GIBA authentication procedure.
- IMS server address (P-CSCF). This hardcoded address may be changed via the provisioning application or via OTA.
- No login/password.

The IMS profile configuration may be changed via the provisioning application or via OTA. From the provisioning application there is a dedicated menu “Generate profile” which permits to select a predefined IMS platform in order to generate an IMS profile associated to this platform. This menu may be useful to quickly generates your IMS profile from your phone number.

If you want to add more IMS platforms in the menu or to change the hardcoded IMS parameters, you should modify the source code of the provisioning application (see class `ProfileProvisioning.java`, method `onCreateDialog`).

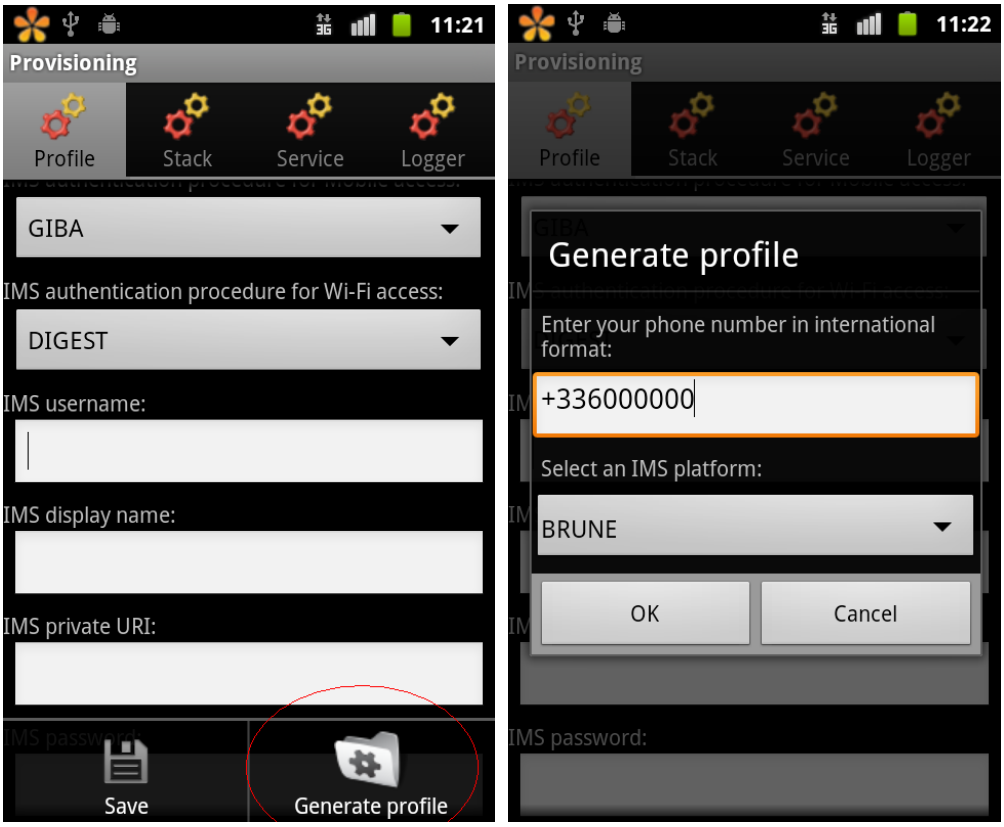


Figure 6: Generate your IMS profile

8. Automatic NAT detection

The RCS-e stack detects automatically if there is a NAT during the registration procedure.

A NAT is detected if the IP address in the Via header of the REGISTER is different than the IP address in the Via header (or Via parameter “received”) of the 200 OK of the REGISTER.

9. Trace management

By default the internal traces of the RCS-e stack are activated with DEBUG level. For production, the trace level should be adapted (e.g. error only).

By default the SIP stack traces are deactivated and may be activated via the provisioning application. These traces concerns only SIP messages and are redirected to SDTOUT.

By default the MSRP media traces are deactivated and may be activated via the provisioning application. These traces concerns only MSRP messages and are redirected to SDTOUT.

The default trace configuration is defined in the class `Logger` and may be changed from the provisioning application.

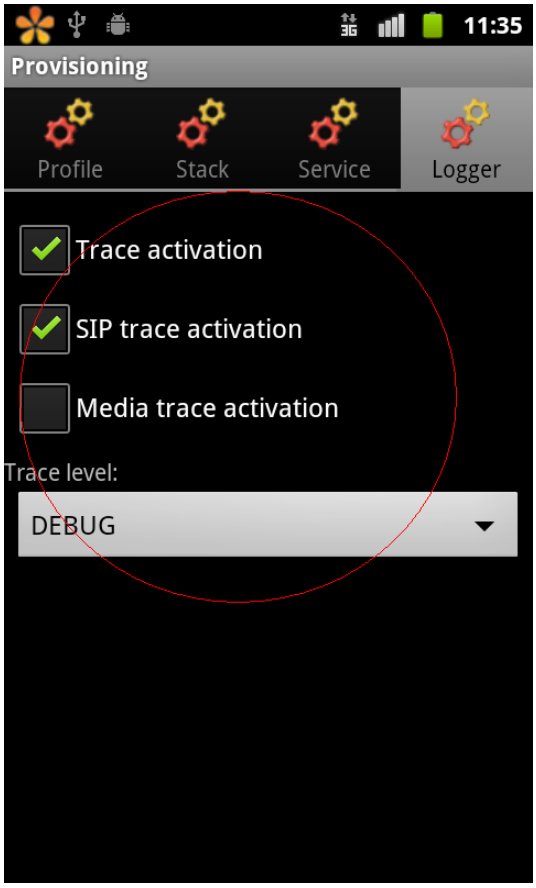


Figure 7: Trace management

10. Display name

A display name may be associated to a SIP-URI or Tel-URI in order to identify a user not yet added in the address book.

By default the display name is empty and should be configured by the end user.

The display name may be changed via the RCS settings application or via the provisioning application (see Profile folder).

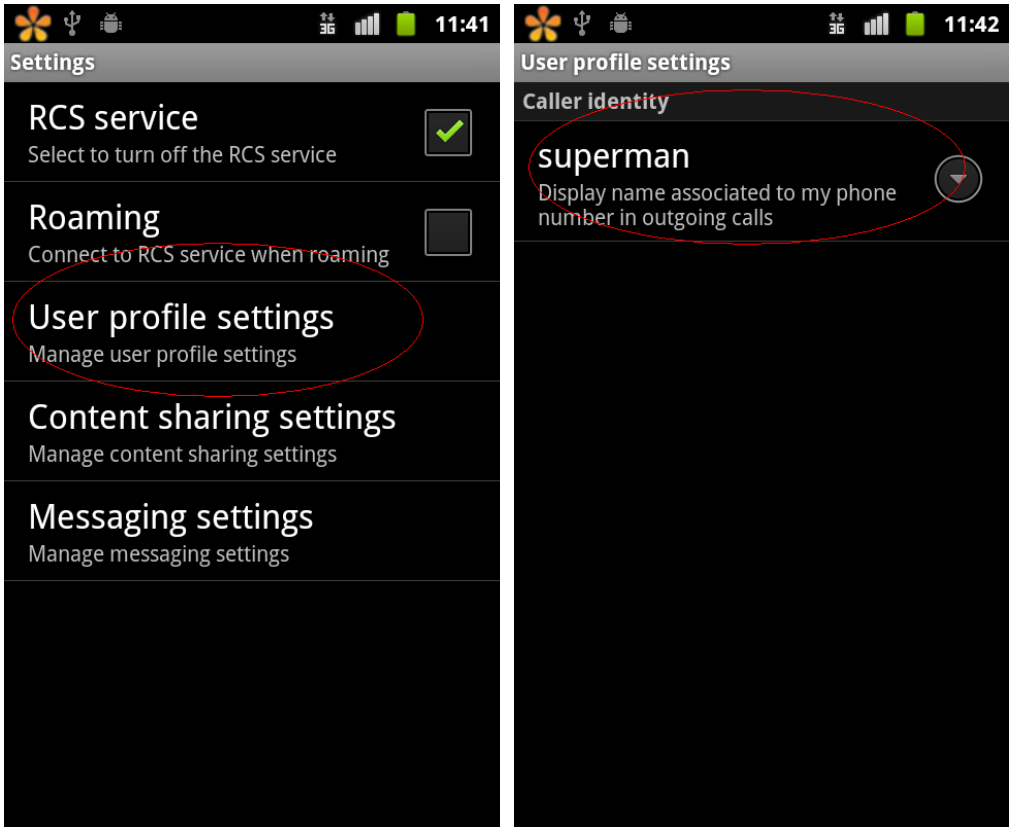


Figure 8: Display name configuration

11. Software release number

The release number is displayed in the following places:

- In the RCS settings application, menu “About”.
- In each SIP messages, see “User-Agent” and “Server” headers.
- In the installed application list from device settings application.

The release number is stored in the following files:

- manifest.xml
- rcs_core_release.xml

12. How to remove the RCS service

The RCS service may be removed by the end user just by deleting its corresponding account from the accounts management application. When the RCS account is removed no more RCS integration exist in the native address book, the RCS background service is automatically stopped.

The RCS service may be reactivated by adding again the RCS account.

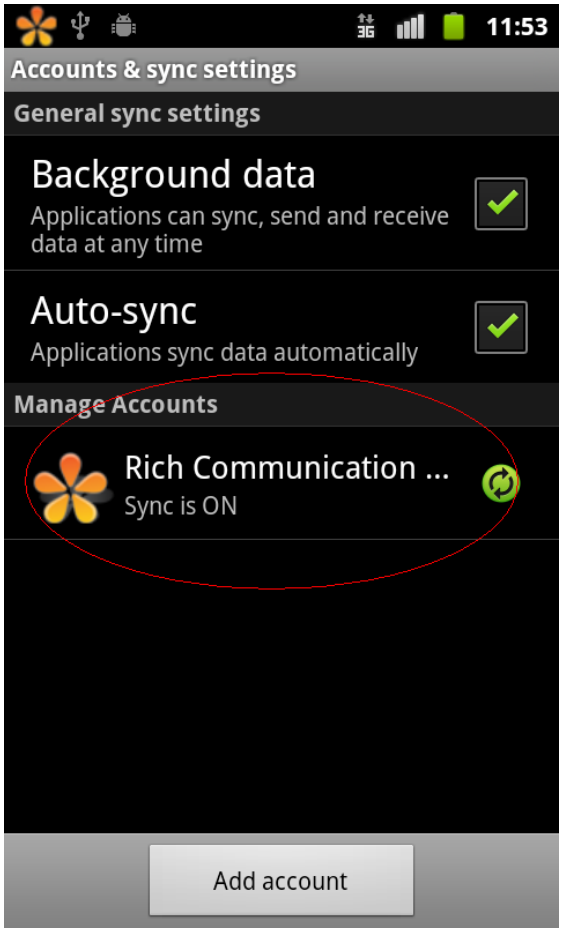


Figure 9: RCS account management

13. How to deactivate the RCS service

The RCS service may be temporarily deactivated by the end user via the RCS settings application.

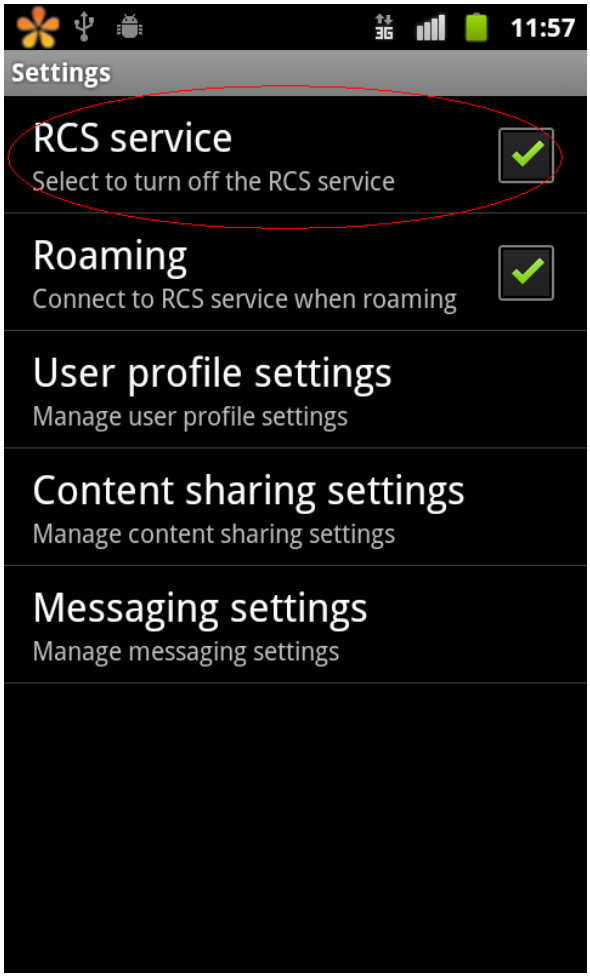


Figure 10: RCS service management

14. RCS permission

Any application (e.g. UI, widget) using the RCS API should:

- Be signed with the same certificate as the RCS stack (API server side).
- Declare the RCS permission in its manifest file:

```
<uses-permission android:name="com.orangelabs.rcs.permission.RCS"/>
```

15. Dynamic RCS extensions

The RCS-e stack discovers dynamically the RCS extensions installed in the device. An RCS extension may be any kind of application or any new IMS service on top of the basic RCS services (IM, File transfer, .etc).

Each extension is prefixed by:

```
urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.orange
```

Samples:

```
urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.orange.game
urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.orange.whiteboard
```

How it works?

1. At the device startup, the stack gets the list of applications supporting the following Intent in their manifest file:

```
<intent-filter>
    <action android:name="com.orangelabs.rcs.capability.EXTENSION"/>
    <data android:mimeType="urn%3Aurn-7%3A3gpp-application.ims.iari.rcse.orange/game"/>
</intent-filter>
```

2. Then the stack shares the discovered extensions with other contacts via the SIP OPTIONS procedure.
3. The end user may know if a given extension or service is supported by other contacts before to invoke its service.
4. If a new extension is installed or removed the stack updates its supported extension automatically and dynamically.

16. Session refresh management

17. AIDL remote exception

```
06-24 13:55:36.140: ERROR/JavaBinder(1834): *** Uncaught remote exception!
(Exceptions are not yet supported across processes.)
06-24 13:55:36.140: ERROR/JavaBinder(1834): java.lang.RuntimeException:
com.orangelabs.rcs.service.api.server.ServerApiException
06-24 13:55:36.140: ERROR/JavaBinder(1834):     at
android.os.Parcel.writeException(Parcel.java:1223)
06-24 13:55:36.140: ERROR/JavaBinder(1834):     at
android.os.Binder.execTransact(Binder.java:322)
06-24 13:55:36.140: ERROR/JavaBinder(1834):     at
dalvik.system.NativeStart.run(Native Method)
06-24 13:55:36.140: ERROR/JavaBinder(1834): Caused by:
com.orangelabs.rcs.service.api.server.ServerApiException
06-24 13:55:36.140: ERROR/JavaBinder(1834):     at
com.orangelabs.rcs.service.api.server.ServerApiUtils.testIms(ServerApiUtils.java:62)
06-24 13:55:36.140: ERROR/JavaBinder(1834):     at
com.orangelabs.rcs.service.api.server.capability.CapabilityApiService.requestCapabilit
ies(CapabilityApiService.java:71)
06-24 13:55:36.140: ERROR/JavaBinder(1834):     at
com.orangelabs.rcs.service.api.client.capability.ICapabilityApi$Stub.onTransact(ICapab
ilityApi.java:53)
06-24 13:55:36.140: ERROR/JavaBinder(1834):     at
android.os.Binder.execTransact(Binder.java:320)
06-24 13:55:36.140: ERROR/JavaBinder(1834):     ... 1 more
```

18. Media players

TO DO

19. RCS API

TO DO

20. Social presence sharing

The presence service or social presence sharing from RCS 2.0 is optional under RCS-e.

The presence service is deactivated by default and may be activated via the provisioning application or via OTA.

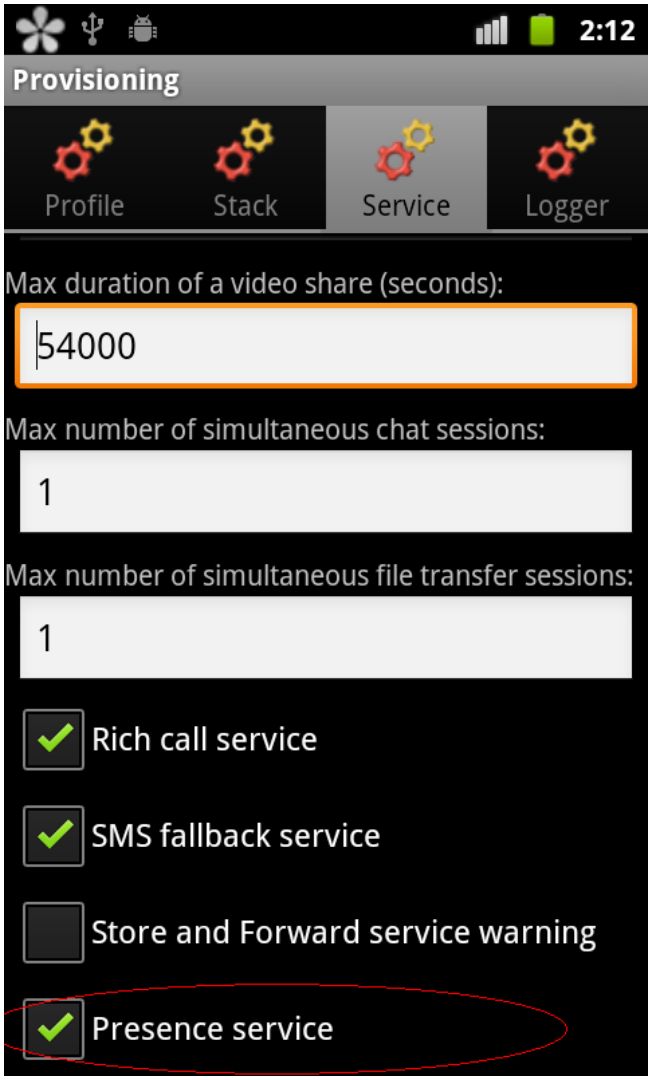


Figure 11: Presence service

21. Phone number format

In the RCS-e stack, all phone numbers are formatted into international format in order to facilitate phone numbers comparison.

By default, all SIP contacts are formatted into Tel-URI. It's possible to force SIP-URI format via the provisioning application or via OTA.

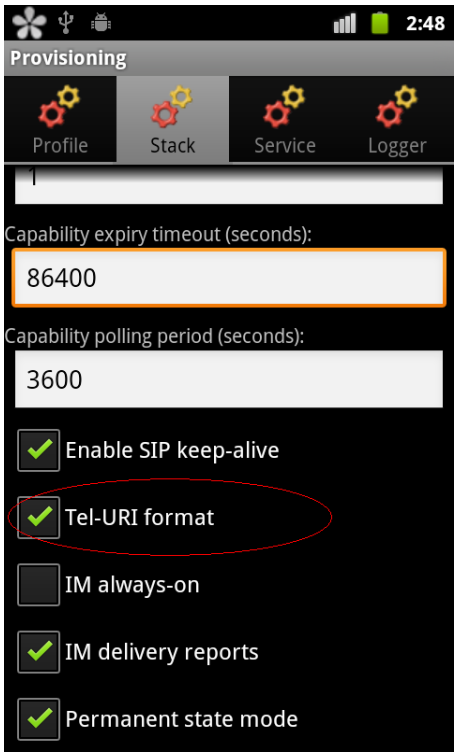


Figure 12: SIP contact format

To format a phone number, the default country code is read from the RCS settings and the default value is "+33 (France)". The country code may be changed via the provisioning application or its hardcoded value changed in the class `RcsSettingsProvider`:

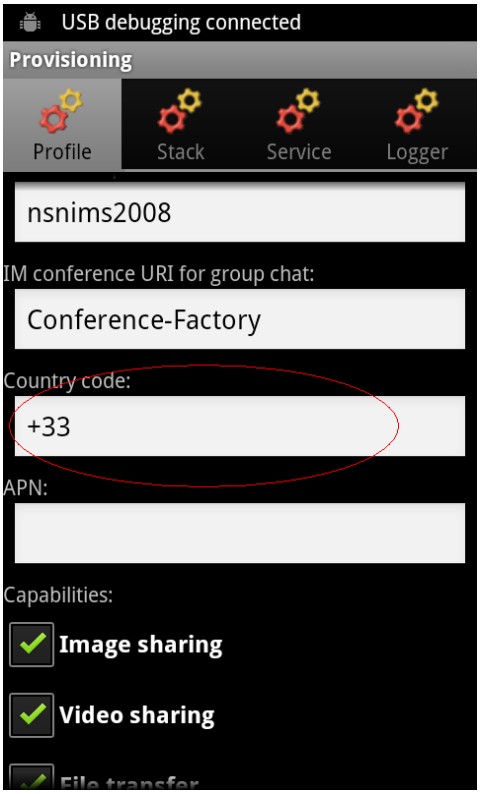


Figure 13: Country code