The Power of Android on Your IP Phone

Agenda

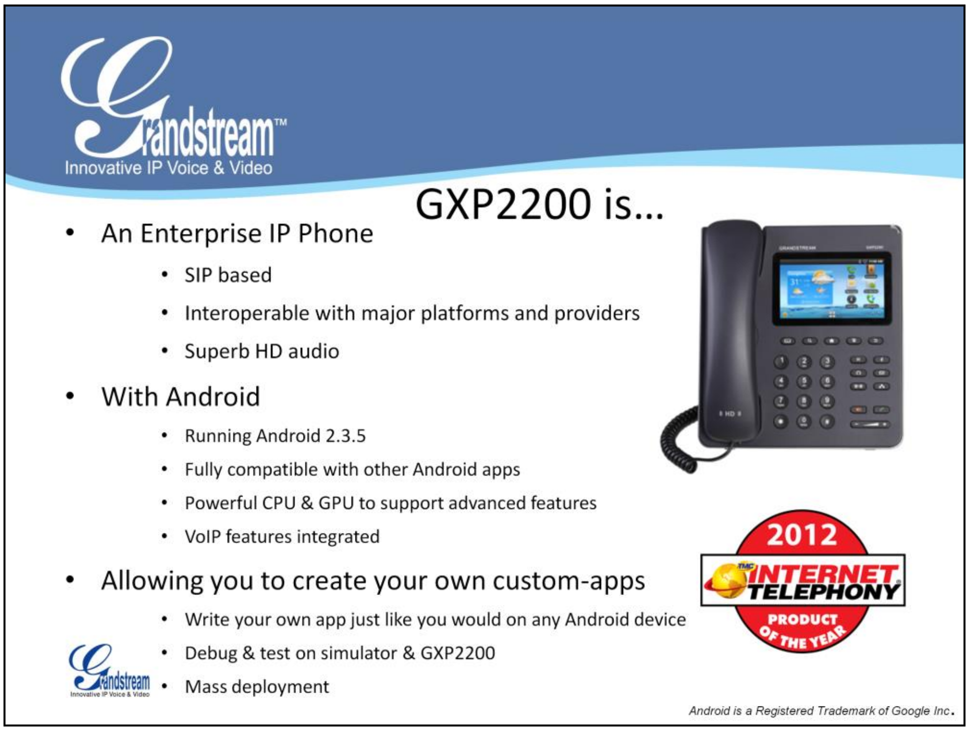
- GXP2200
- Android Platform Overview
- Android Application Development
- Grandstream Enhanced API
- App Development Flow on GXP2200
- GMI/Web Service API
- GUI Customization

The Power of Android on your Desktop

THE POWER OF ANDROID™ IS NOW ON YOUR DESKTOP
Grandstream
Innovative IP Voice & Video
GXP2200 ENTERPRISE APPLICATION PHONE FOR ANDROID
ORDER NOW!

- One of just a handful of Android-based IP Phones in the world
  - One of even fewer that runs an OPEN version of Android
- Running Android 2.3 Gingerbread, which is currently the most widely-used version of Android

Android is a Registered Trademark of Google Inc.



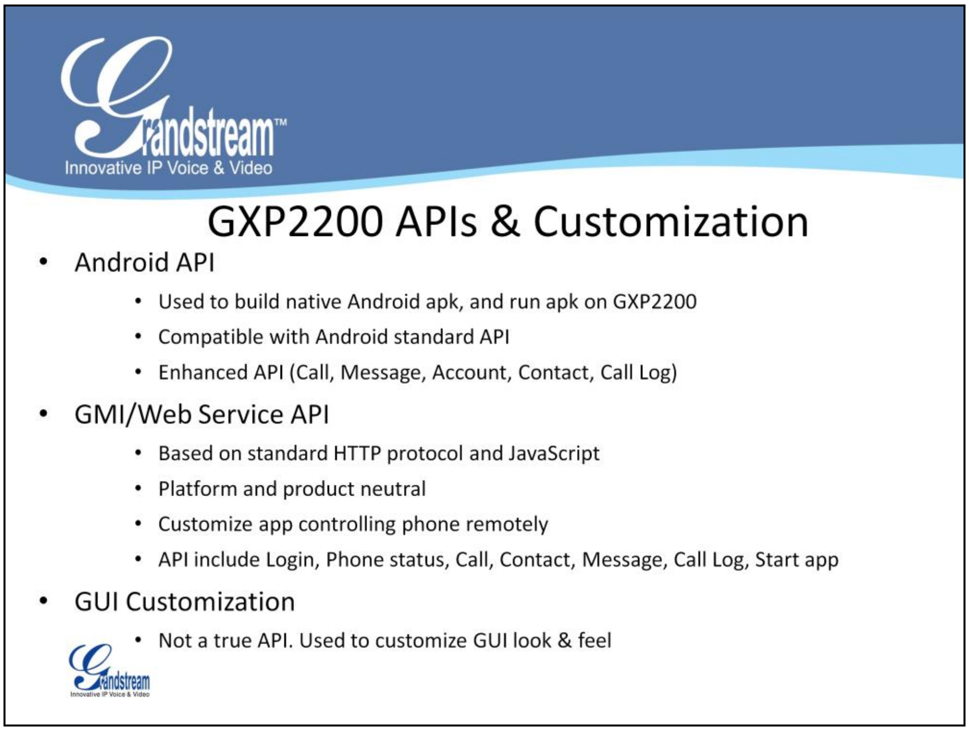
GXP2200 is…

- An Enterprise IP Phone
  - SIP based
  - Interoperable with major platforms and providers
  - Superb HD audio
- With Android
  - Running Android 2.3.5
  - Fully compatible with other Android apps
  - Powerful CPU & GPU to support advanced features
  - VoIP features integrated
- Allowing you to create your own custom-apps
  - Write your own app just like you would on any Android device
  - Debug & test on simulator & GXP2200
  - Mass deployment

Android is a Registered Trademark of Google Inc.

-Android 2.3 is the most used version of Android on the market today with almost 60% of all Android devices running 2.3 Gingerbread

-Winner of Internet Telephony Magazine’s 2012 Product of the Year Award



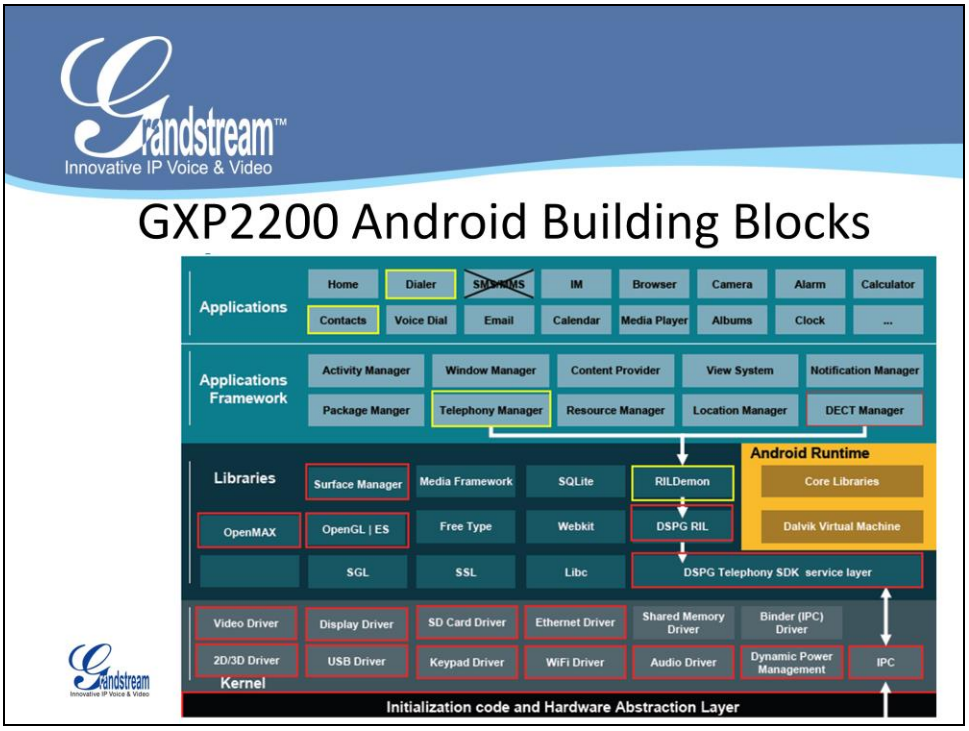
GXP2200 APIs & Customization

- Android API
  - Used to build native Android apk, and run apk on GXP2200
  - Compatible with Android standard API
  - Enhanced API (Call, Message, Account, Contact, Call Log)
- GMI/Web Service API
  - Based on standard HTTP protocol and JavaScript
  - Platform and product neutral
  - Customize app controlling phone remotely
  - API include Login, Phone status, Call, Contact, Message, Call Log, Start app
- GUI Customization
  - Not a true API. Used to customize GUI look & feel

There are 3 ways to enhance, expand and customize GXP2200

using Android API to develop your own app, using GMI to develop web service based tools, or simply use config tool to customize the look & feel of the phone

These 3 methods are independent of each other. They can be used together with each other, or separately



GXP2200 is based on Android 2.3, aka Gingerbread. It has similar building blocks of typical android devices, the low level device drivers, the middle-tier including Davik runtime environment, 2D/3D acceleration, video encoding/decoding acceleration and upper layer application framework

There is one major difference though. Android 2.3 or even the latest 4.x today is not centered around VoIP. It's primary targeted usage is baseband cellular

Grandstream has gone through great length to take out the pieces the are only relevant to baseband cellular, and replace them with VoIP functionalities, e.g. dialer, contact, telephony manager etc.

The main challenge here is to ensure the quality of service and minimize delay caused by Android framework

All these changes and details are hidden inside the framework and encapsulated by the API.

Innovative IP Voice & Video

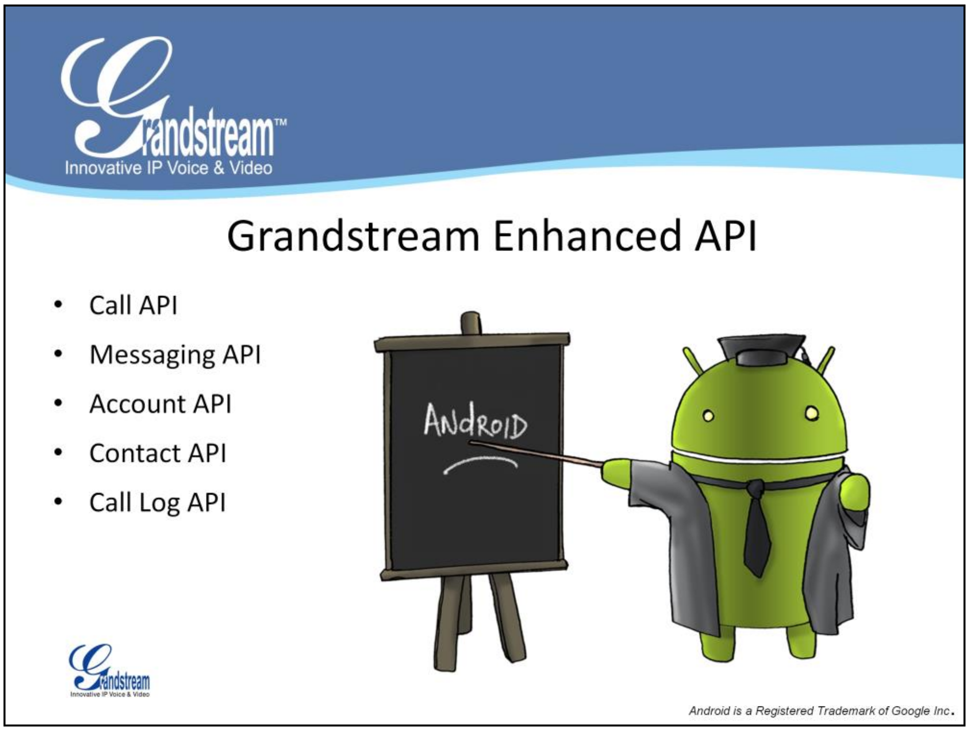
GXP2200 APIs & Customization

1. Android API

2. GMI/Web Service API

3. GUI Customization

Innovative IP Voice & Video



Grandstream Enhanced API

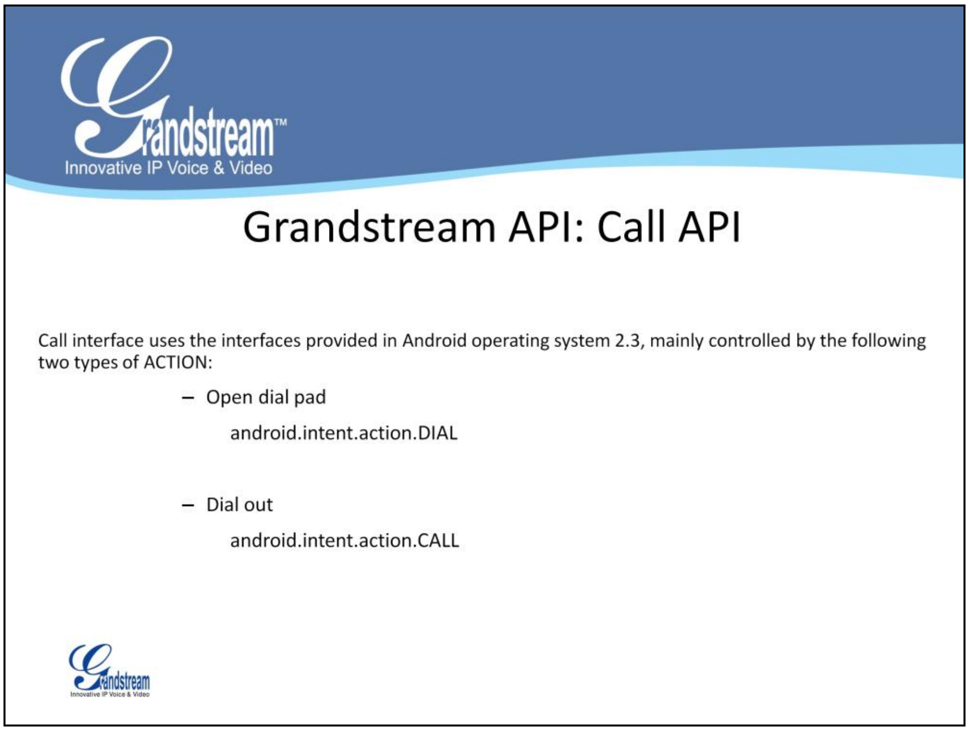
- Call API
- Messaging API
- Account API
- Contact API
- Call Log API

Android is a Registered Trademark of Google Inc.

Grandstream enhanced API is based on standard Android API with additional functionalities and is customized for VoIP

Native Android API can be used with no restriction

There are 5 categories of the API



Grandstream API: Call API

Call interface uses the interfaces provided in Android operating system 2.3, mainly controlled by the following two types of ACTION:

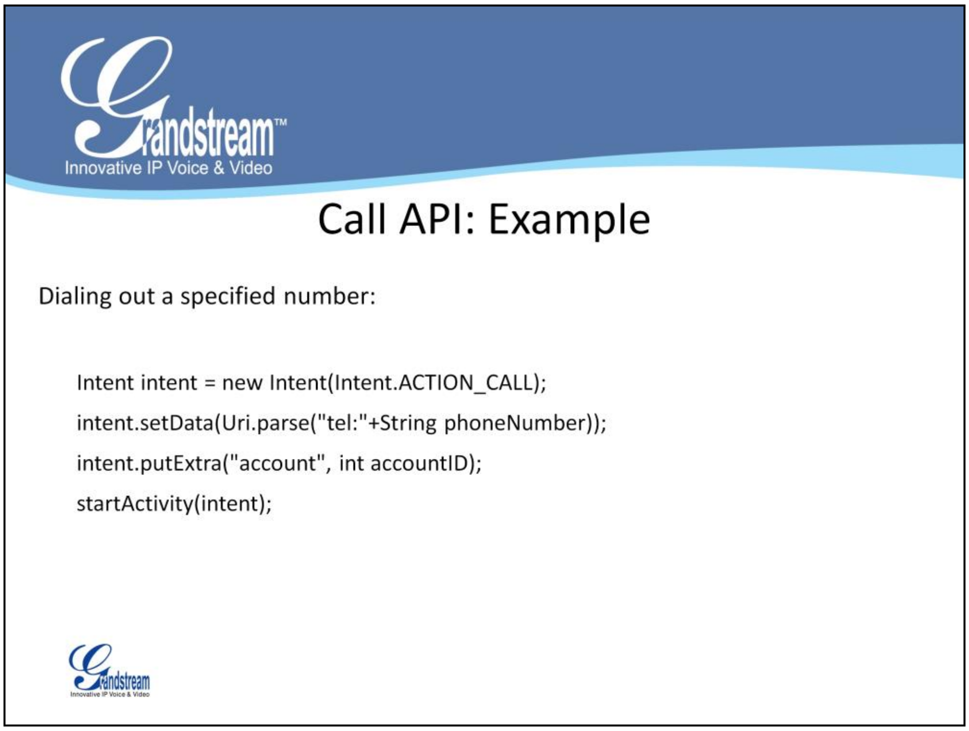
- Open dial pad

  android.intent.action.DIAL

- Dial out

  android.intent.action.CALL

The call API is based on Android Call interface. It can be used to open dial pad, edit dialing numbers, dial a number or redial

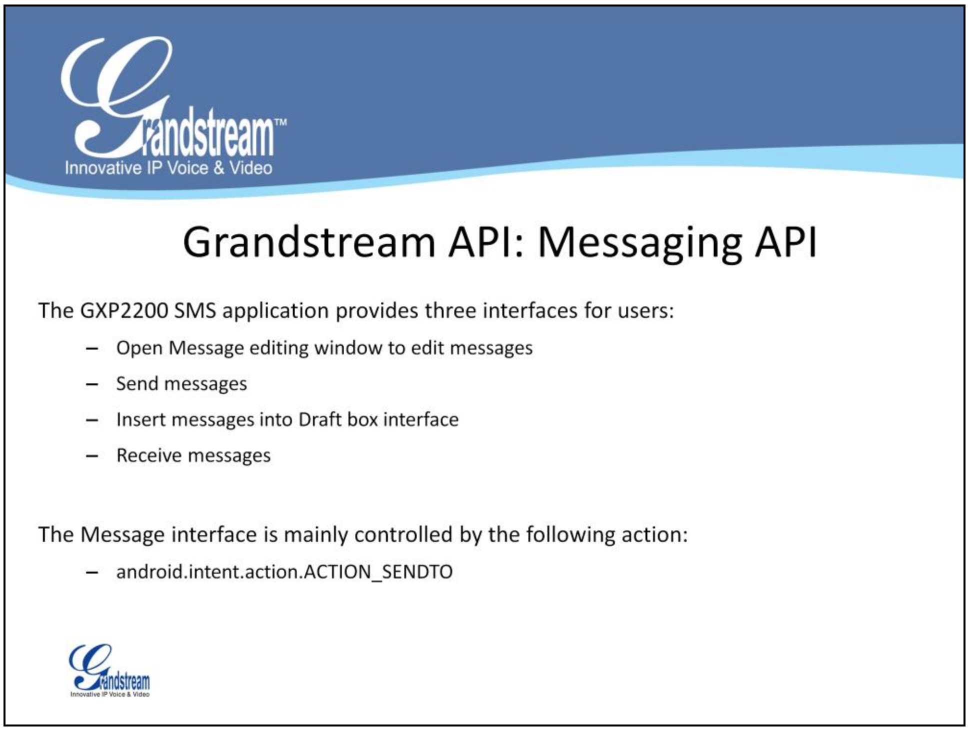


Call API: Example

Dialing out a specified number:

```
Intent intent = new Intent(Intent.ACTION_CALL);
intent.setData(Uri.parse("tel:"+String phoneNumber));
intent.putExtra("account", int accountID);
startActivity(intent);
```

The example here will open the dial pad, display the phone number and specify the account that is used



Grandstream API: Messaging API

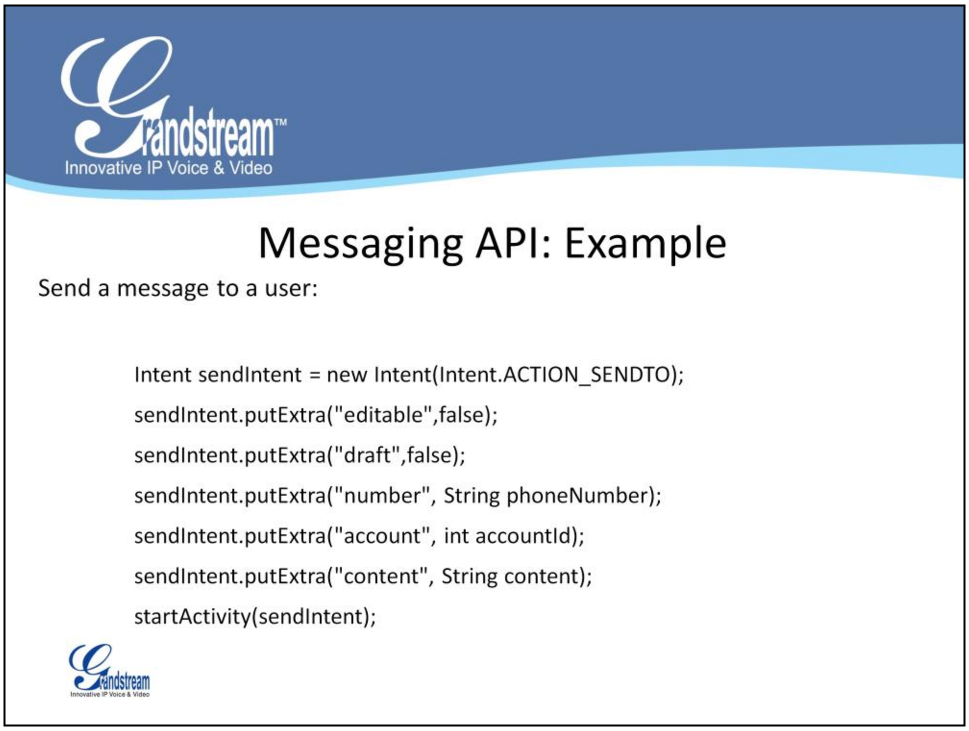
The GXP2200 SMS application provides three interfaces for users:

- Open Message editing window to edit messages
- Send messages
- Insert messages into Draft box interface
- Receive messages

The Message interface is mainly controlled by the following action:

- android.intent.action.ACTION_SENDTO

Grandstream messaging API is based on Android SMS API

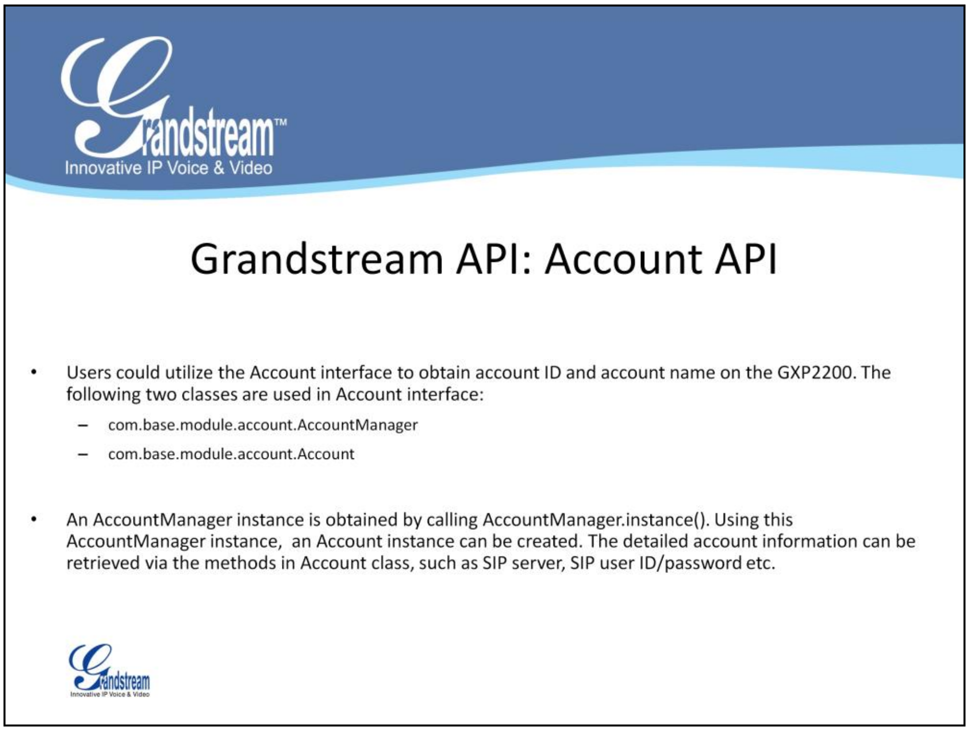


Messaging API: Example

Send a message to a user:

```
Intent sendIntent = new Intent(Intent.ACTION_SENDTO);
sendIntent.putExtra("editable",false);
sendIntent.putExtra("draft",false);
sendIntent.putExtra("number", String phoneNumber);
sendIntent.putExtra("account", int accountId);
sendIntent.putExtra("content", String content);
startActivity(sendIntent);
```

This example shows how to send a SIP message to the specified number

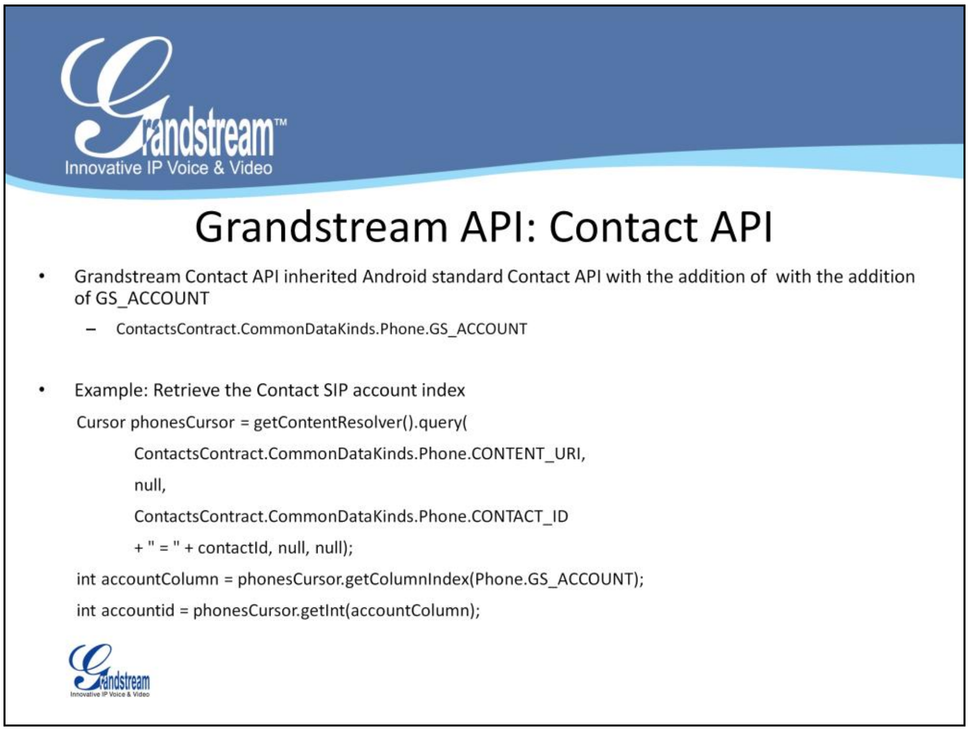


Grandstream API: Account API

- Users could utilize the Account interface to obtain account ID and account name on the GXP2200. The following two classes are used in Account interface:
  - com.base.module.account.AccountManager
  - com.base.module.account.Account
- An AccountManager instance is obtained by calling AccountManager.instance(). Using this AccountManager instance, an Account instance can be created. The detailed account information can be retrieved via the methods in Account class, such as SIP server, SIP user ID/password etc.

Grandstream Account API can be used to get and set account related information, such as account ID, name, SIP server, outbound proxy, SIP user ID, password, display name etc

Similar to Android Account API, before using this API, an AccountManager instance has to be obtained first



Grandstream API: Contact API

- Grandstream Contact API inherited Android standard Contact API with the addition of with the addition of GS_ACCOUNT
  - ContactsContract.CommonDataKinds.Phone.GS_ACCOUNT

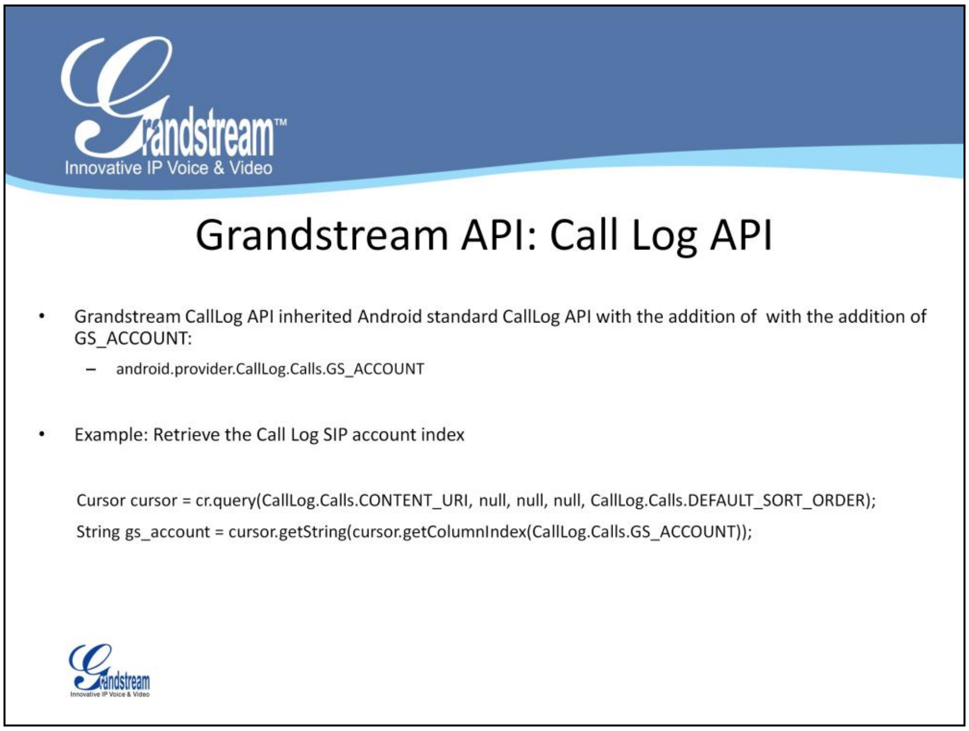
- Example: Retrieve the Contact SIP account index

```
Cursor phonesCursor = getContentResolver().query(
        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
        null,
        ContactsContract.CommonDataKinds.Phone.CONTACT_ID
        + " = " + contactId, null, null);
int accountColumn = phonesCursor.getColumnIndex(Phone.GS_ACCOUNT);
int accountid = phonesCursor.getInt(accountColumn);
```

Grandstream Contact API is almost identical to the standard Android Contact API, with the addition of SIP account concept

Developers can search the phone contact database with this API

The API can also be used to edit, insert or delete entries in the contact



Grandstream API: Call Log API

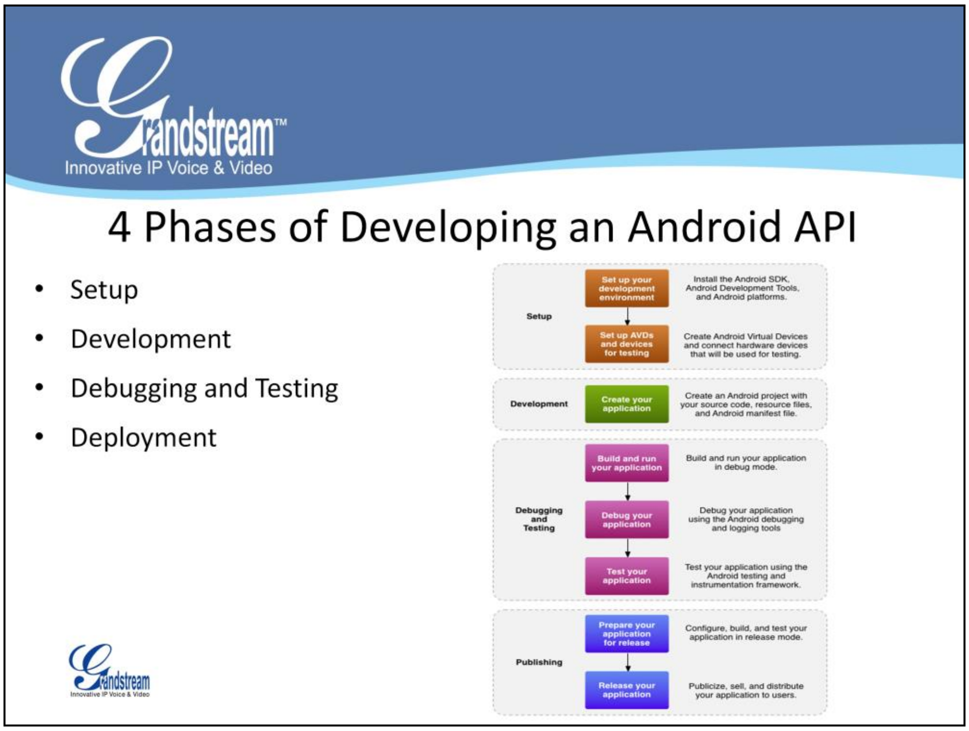
- Grandstream CallLog API inherited Android standard CallLog API with the addition of with the addition of GS_ACCOUNT:
  - android.provider.CallLog.Calls.GS_ACCOUNT
- Example: Retrieve the Call Log SIP account index

```
Cursor cursor = cr.query(CallLog.Calls.CONTENT_URI, null, null, null, CallLog.Calls.DEFAULT_SORT_ORDER);
String gs_account = cursor.getString(cursor.getColumnIndex(CallLog.Calls.GS_ACCOUNT));
```

Again, Grandstream Call Log API is almost identical to Android standard Call Log API, with the addition of SIP account concept

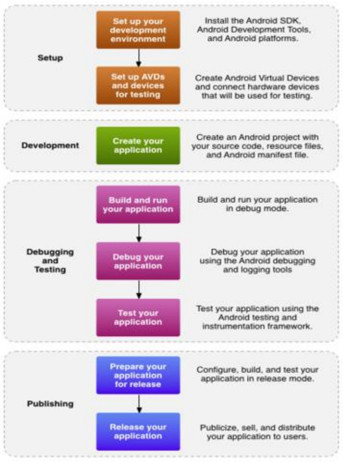
Developers can search the call log database with this API

It can also be used to delete entries in Call Log



4 Phases of Developing an Android API

- Setup
- Development
- Debugging and Testing
- Deployment



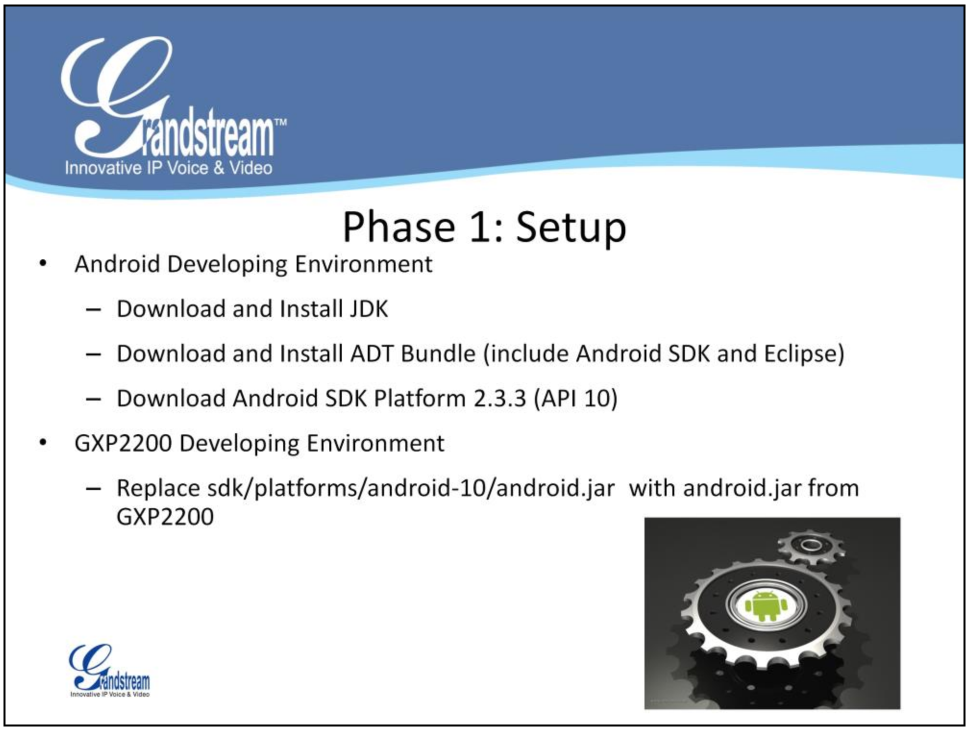
Setup
Set up your development environment — Install the Android SDK, Android Development Tools, and Android platforms.
Set up AVDs and devices for testing — Create Android Virtual Devices and connect hardware devices that will be used for testing.

Development
Create your application — Create an Android project with your source code, resource files, and Android manifest file.

Debugging and Testing
Build and run your application — Build and run your application in debug mode.
Debug your application — Debug your application using the Android debugging and logging tools
Test your application — Test your application using the Android testing and instrumentation framework.

Publishing
Prepare your application for release — Configure, build, and test your application in release mode.
Release your application — Publicize, sell, and distribute your application to users.

Just like any native Android app development, developing GXP2200 Android app usually goes through 4 phases



Phase 1: Setup

- Android Developing Environment
  - Download and Install JDK
  - Download and Install ADT Bundle (include Android SDK and Eclipse)
  - Download Android SDK Platform 2.3.3 (API 10)
- GXP2200 Developing Environment
  - Replace sdk/platforms/android-10/android.jar with android.jar from GXP2200

The first phase is to setup the development environment.

Android uses Java as the programming language.

The first several steps are identical to typical Android app development:

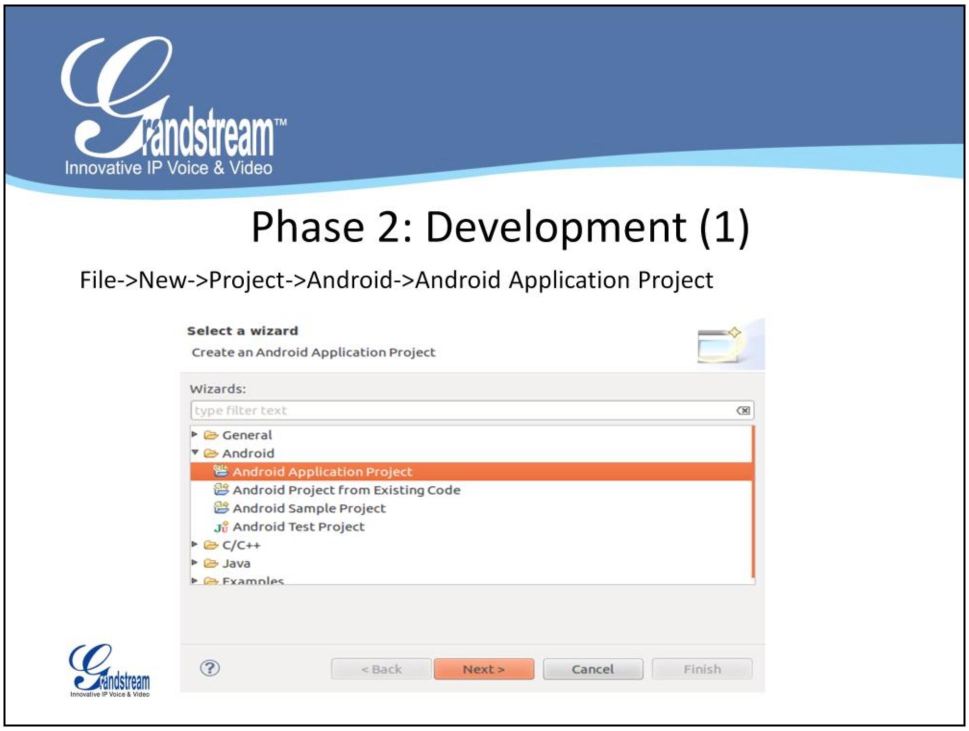
download and install Java Development Toolkit

download and install Android Developer Tools including Android SDK and Eclipse. Eclipse is a very popular cross-platform IDE. Eclipse is used for C/C++ development as well

Download Android SDK Platform 2.3.3

These software packages are available from Android developer website

The only difference, comparing to standard Android development, is that developers need to replace android.jar with the GXP2200 version, which contains Grandstream's enhancements and innovations to VoIP in Android



Phase 2: Development (1)

File->New->Project->Android->Android Application Project

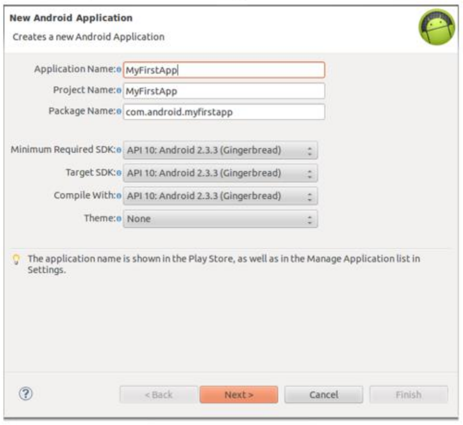
The phase 2 is the development of the Android app, implementing your business logics, providing value-add services etc.

The screen capture here shows the new project wizard in Eclipse. Remember Eclipse is the integrated development environment that has the support for development and debugging with a single, easy to use graphic interface

Here developers need to chose Android Application Project

Development (2)

Name your application, select API 10, then click 'Next'



New Android Application

Creates a new Android Application

Application Name: MyFirstApp

Project Name: MyFirstApp

Package Name: com.android.myfirstapp

Minimum Required SDK: API 10: Android 2.3.3 (Gingerbread)

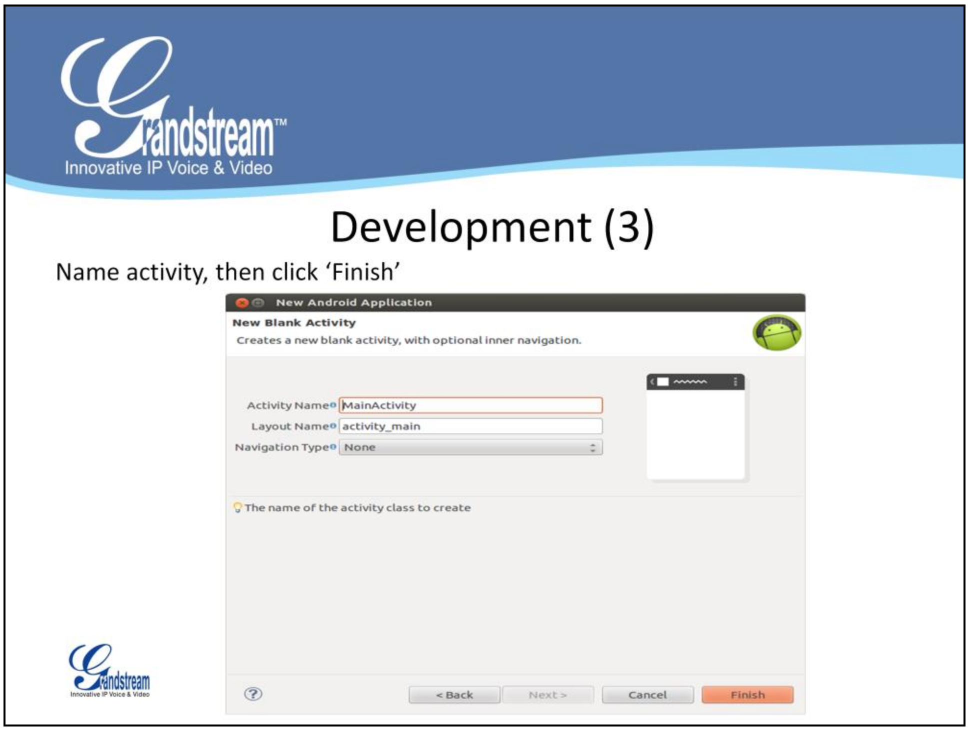
Target SDK: API 10: Android 2.3.3 (Gingerbread)

Compile With: API 10: Android 2.3.3 (Gingerbread)

Theme: None

The application name is shown in the Play Store, as well as in the Manage Application list in Settings.

The next step is to name the project and select the API version for the project. GXP2200 is running on Android 2.3.5



Development (3)

Name activity, then click 'Finish'

New Android Application

New Blank Activity

Creates a new blank activity, with optional inner navigation.

Activity Name	MainActivity
Layout Name	activity_main
Navigation Type	None

The name of the activity class to create

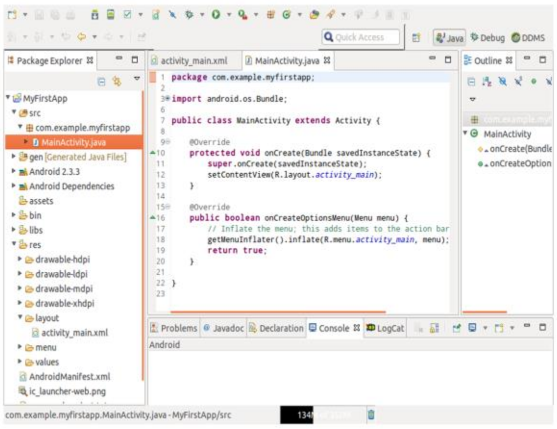
The next step is to name the activity. An activity is conceptually similar to a window in Windows

It deals with user interactions and provide proper life cycle management, such as onCreate, onPause, OnRusume, OnDestroy etc

User interactions are centered around Android activity

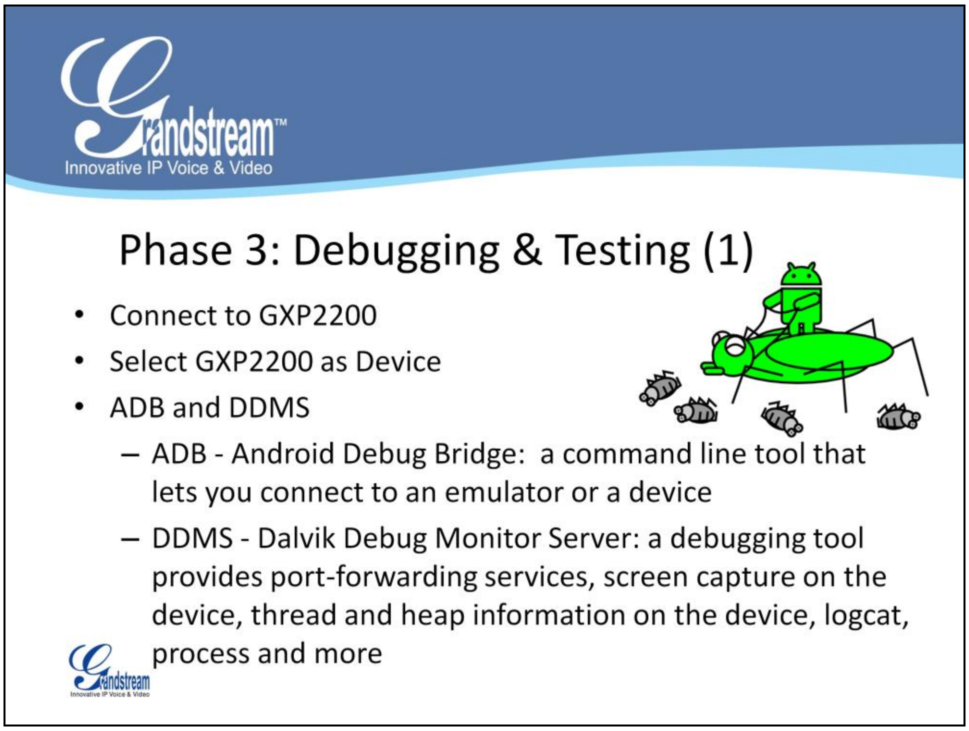
Development (4)

New project is created



```
package com.example.myfirstapp;

import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }

}
```

After clicking the finish button, the new project is created and the skeleton code is automatically generated



Phase 3: Debugging & Testing (1)

- Connect to GXP2200
- Select GXP2200 as Device
- ADB and DDMS
  - ADB - Android Debug Bridge: a command line tool that lets you connect to an emulator or a device
  - DDMS - Dalvik Debug Monitor Server: a debugging tool provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process and more

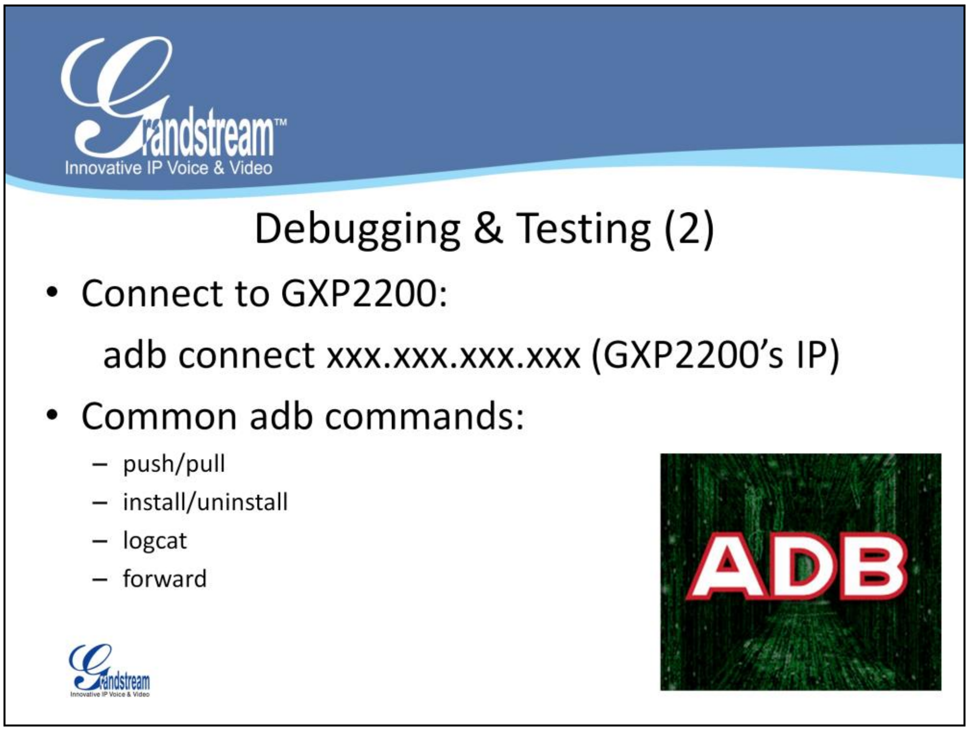
After the initial coding is completed, developers need to debug and test the business logic just implemented

adb and DDMS are the two mostly used debugging tools in Android. GXP2200 is no exception

adb is a command line tool that allows developers to connect to an emulator or a device

However, most developers use ddms which is seamlessly integrated into Eclipse. Developers can issue different debugging commands via Eclipse GUI, including setting breakpoints and evaluating variables

To use the debugger in Eclipse, developers need to connect to GXP2200 first

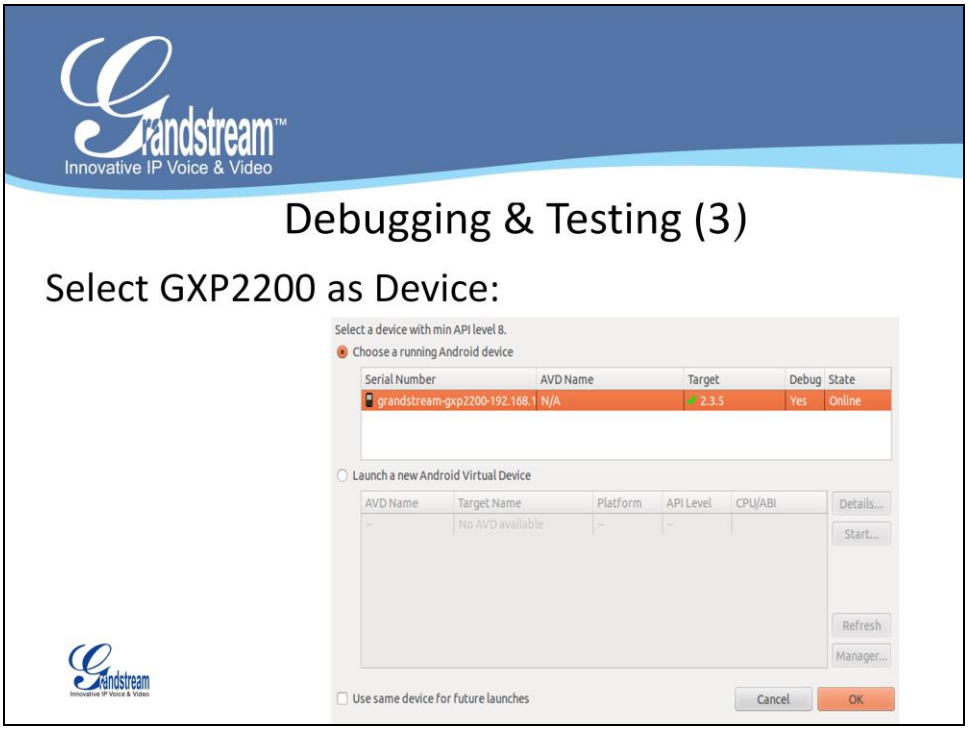


Debugging & Testing (2)

- Connect to GXP2200:

  adb connect xxx.xxx.xxx.xxx (GXP2200's IP)
- Common adb commands:
  - push/pull
  - install/uninstall
  - logcat
  - forward

Developers use adb connect to connect to GXP2200 via TCP/IP



Debugging & Testing (3)

Select GXP2200 as Device:

Select a device with min API level 8.

(●) Choose a running Android device

Serial Number	AVD Name	Target	Debug	State
grandstream-gxp2200-192.168.1	N/A	✓ 2.3.5	Yes	Online

( ) Launch a new Android Virtual Device

AVD Name	Target Name	Platform	API Level	CPU/ABI
--	No AVD available	--	--	

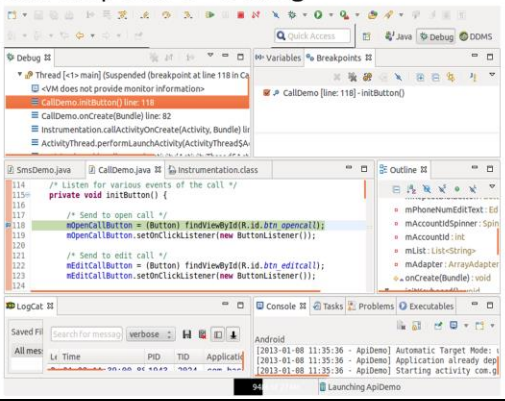
☐ Use same device for future launches

Then select GXP2200 as the target device

Debugging & Testing (4)

Debug and DDMS:

(1) Eclipse: Window->Open Perspective->Debug



```
Thread [<1> main] (Suspended (breakpoint at line 118 in Ca
    <VM does not provide monitor information>
    CallDemo.initButton() line: 118
    CallDemo.onCreate(Bundle) line: 82
    Instrumentation.callActivityOnCreate(Activity, Bundle) lir
    ActivityThread.performLaunchActivity(ActivityThread$A
```

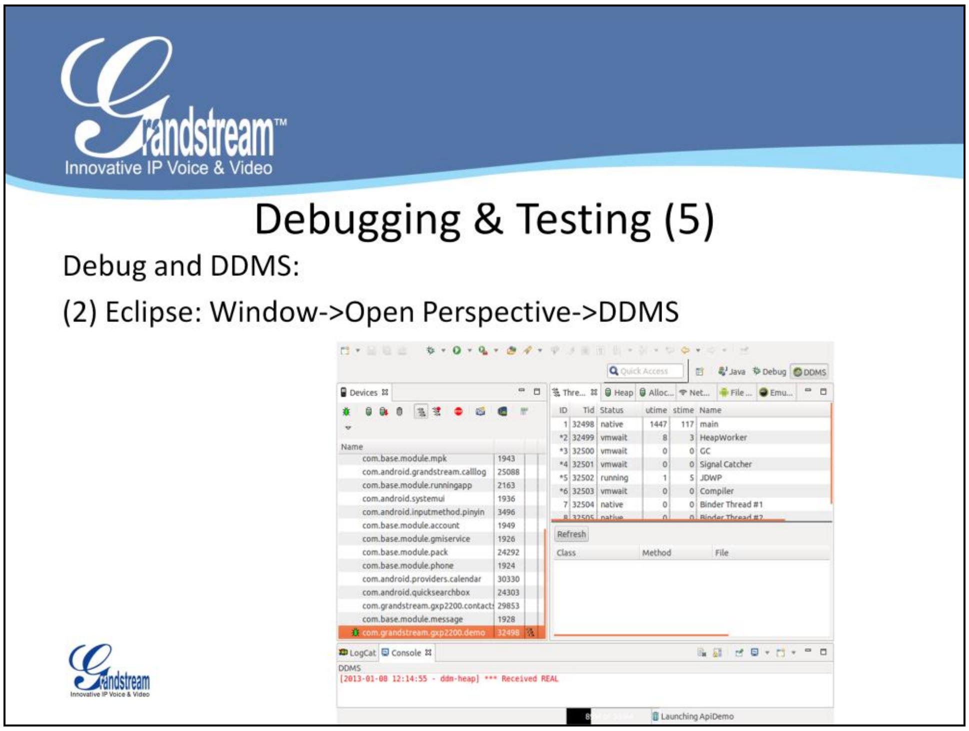
```
CallDemo [line: 118] - initButton()
```

```
114     /* Listen for various events of the call */
115     private void initButton() {
116
117         /* Send to open call */
118         mOpenCallButton = (Button) findViewById(R.id.btn_opencall);
119         mOpenCallButton.setOnClickListener(new ButtonListener());
120
121         /* Send to edit call */
122         mEditCallButton = (Button) findViewById(R.id.btn_editcall);
123         mEditCallButton.setOnClickListener(new ButtonListener());
124
```

```
mPhoneNumEditText : Ed
mAccountIdSpinner : Spin
mAccountId : int
mList : List<String>
mAdapter : ArrayAdapter
onCreate(Bundle) : void
```

```
Android
[2013-01-08 11:35:36 - ApiDemo] Automatic Target Mode: u
[2013-01-08 11:35:36 - ApiDemo] Application already dep
[2013-01-08 11:35:36 - ApiDemo] Starting activity com.g
```

The debugging interface is very similar to traditional C/C++ debugging tools: setting break points, single step, evaluating variables etc



Grandstream
Innovative IP Voice & Video

Debugging & Testing (5)

Debug and DDMS:

(2) Eclipse: Window->Open Perspective->DDMS

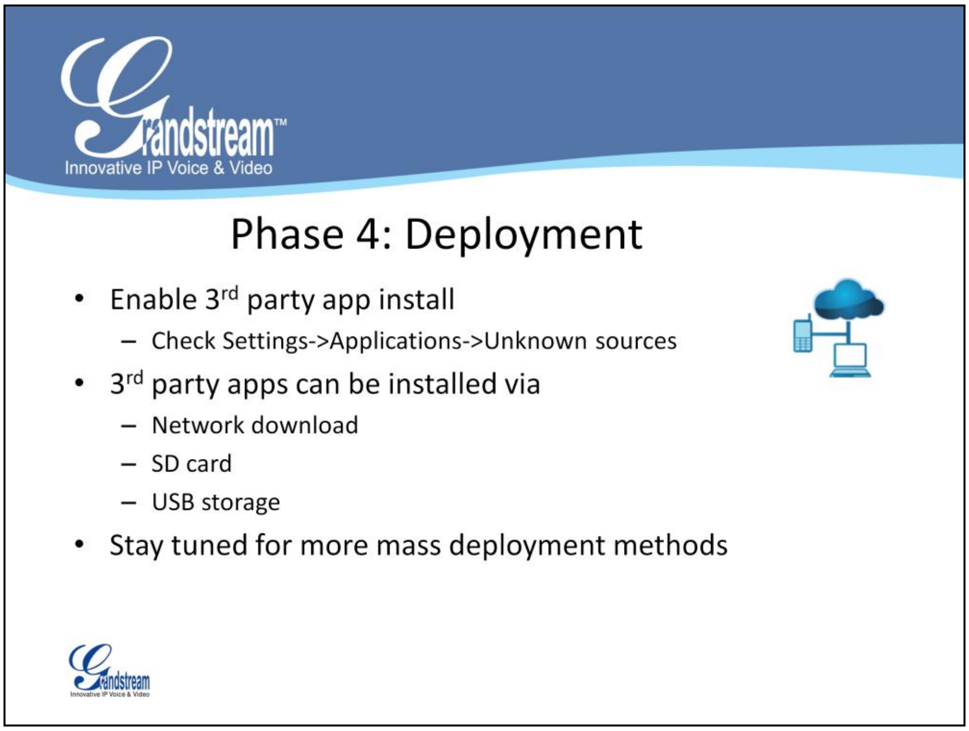
Name	
com.base.module.mpk	1943
com.android.grandstream.calllog	25088
com.base.module.runningapp	2163
com.android.systemui	1936
com.android.inputmethod.pinyin	3496
com.base.module.account	1949
com.base.module.gmiservice	1926
com.base.module.pack	24292
com.base.module.phone	1924
com.android.providers.calendar	30330
com.android.quicksearchbox	24303
com.grandstream.gxp2200.contacts	29853
com.base.module.message	1928
com.grandstream.gxp2200.demo	32498

ID	Tid	Status	utime	stime	Name
1	32498	native	1447	117	main
*2	32499	vmwait	8	3	HeapWorker
*3	32500	vmwait	0	0	GC
*4	32501	vmwait	0	0	Signal Catcher
*5	32502	running	1	5	JDWP
*6	32503	vmwait	0	0	Compiler
7	32504	native	0	0	Binder Thread #1
8	32505	native	0	0	Binder Thread #2

Class	Method	File

```
DDMS
[2013-01-08 12:14:55 - ddm-heap] *** Received REAL
```

Developers can also check running processes, read console logs

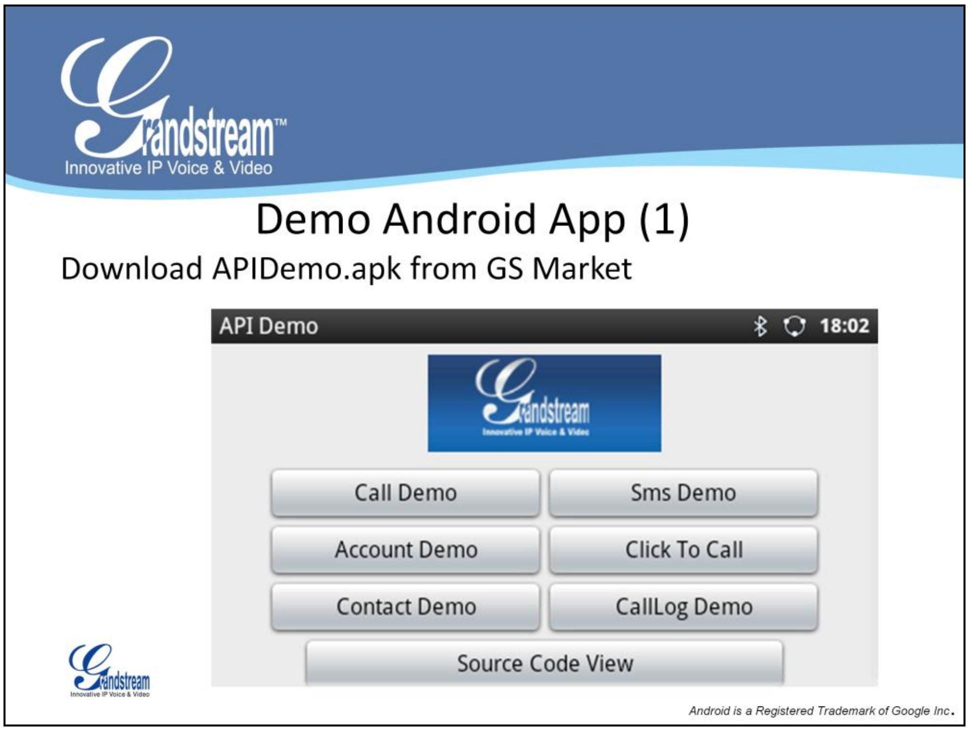


Innovative IP Voice & Video

Phase 4: Deployment

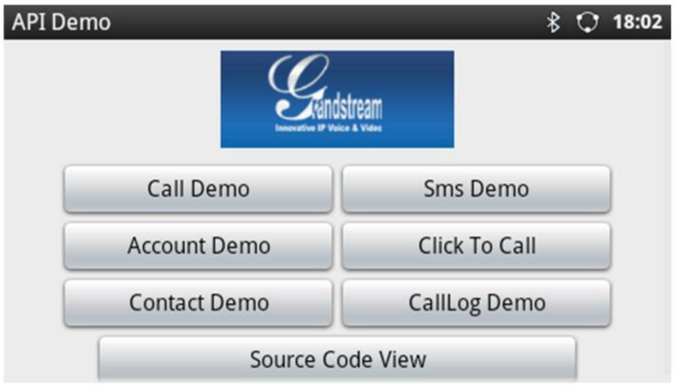
- Enable 3rd party app install
  - Check Settings->Applications->Unknown sources
- 3rd party apps can be installed via
  - Network download
  - SD card
  - USB storage
- Stay tuned for more mass deployment methods

Innovative IP Voice & Video



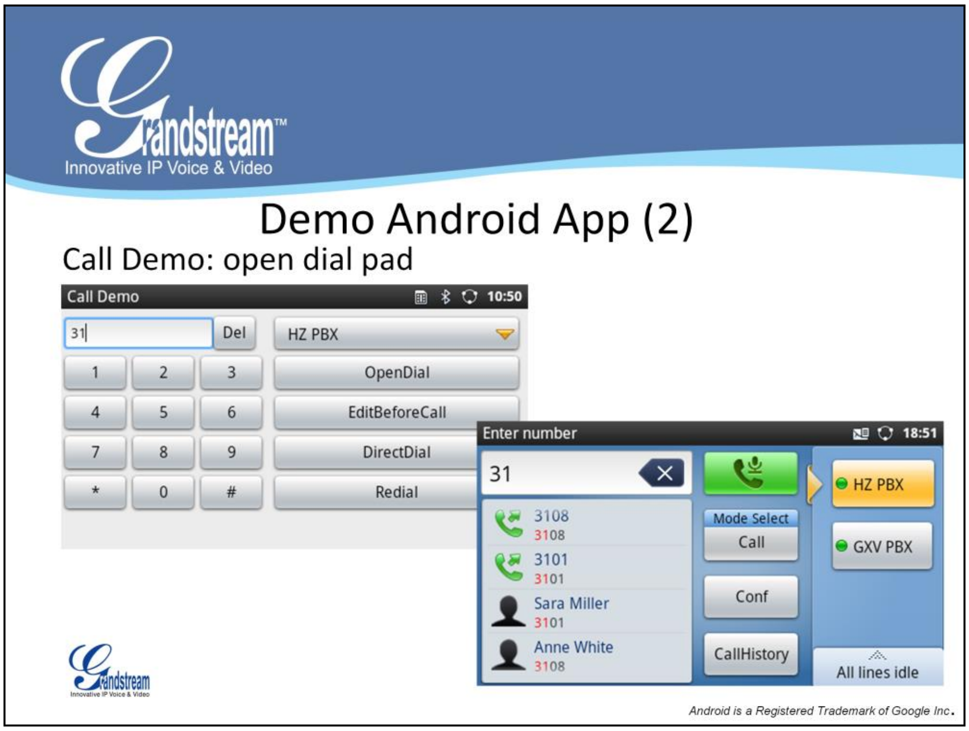
Demo Android App (1)

Download APIDemo.apk from GS Market



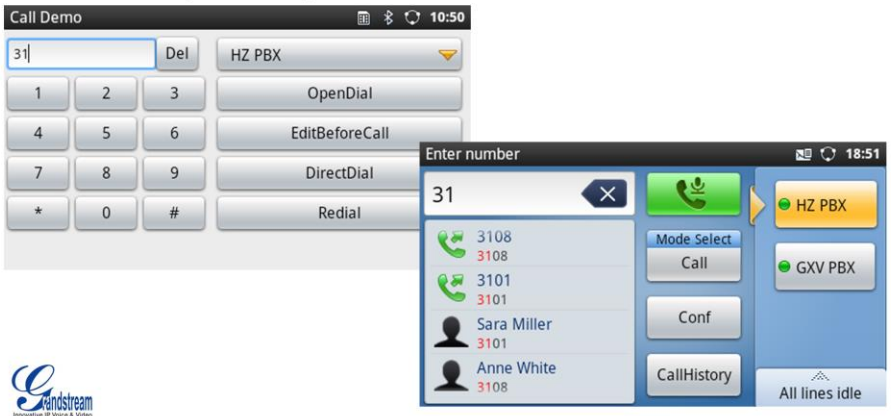
Android is a Registered Trademark of Google Inc.

The demo app package can be downloaded from Grandstream Market Place, with source code



Demo Android App (2)

Call Demo: open dial pad



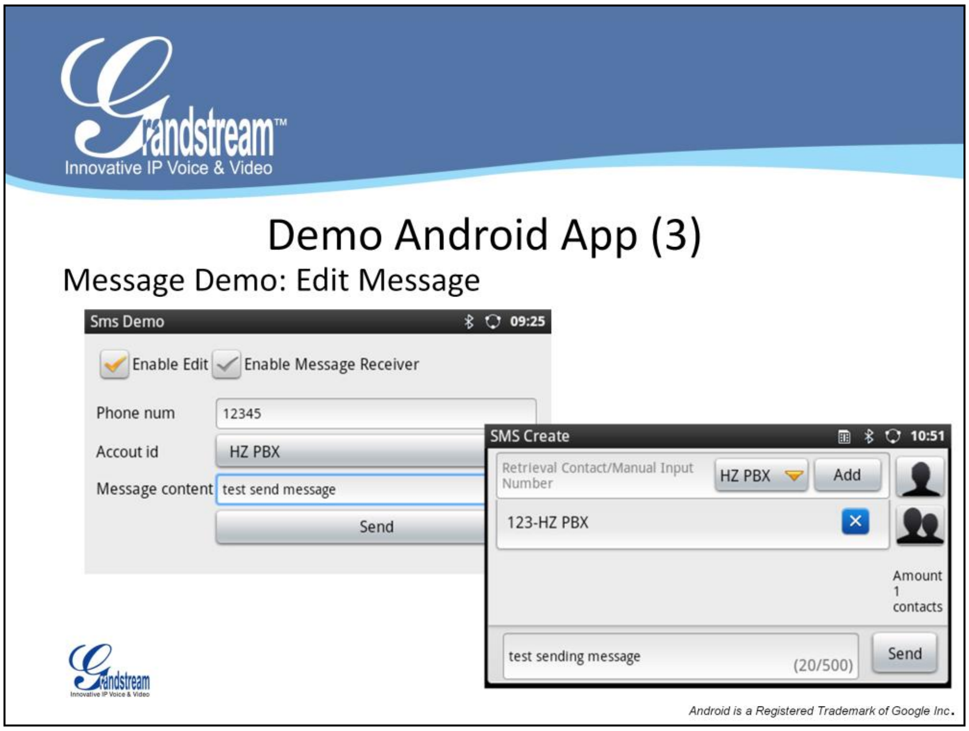
Android is a Registered Trademark of Google Inc.

This is a demo apk using Call API

It displays the customized dial pad on top left

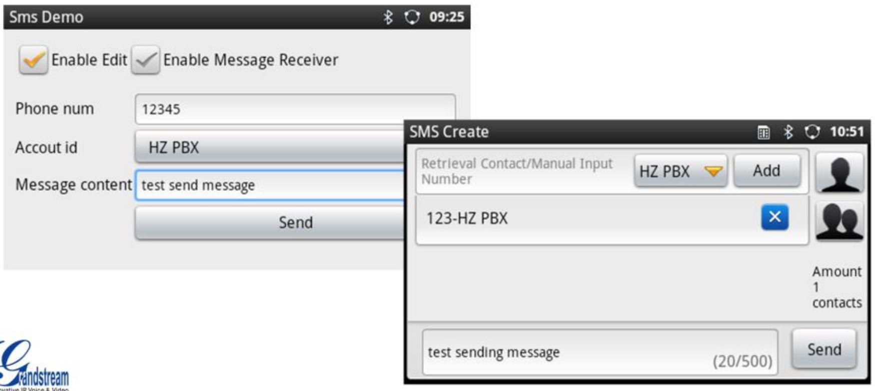
Clicking on OpenDial or EditBeforeCall will invoke the GXP2200 built-in dial pad as shown on bottom right

Users can also click DirectDial to make a call and go directly to call interface



Demo Android App (3)

Message Demo: Edit Message



Android is a Registered Trademark of Google Inc.

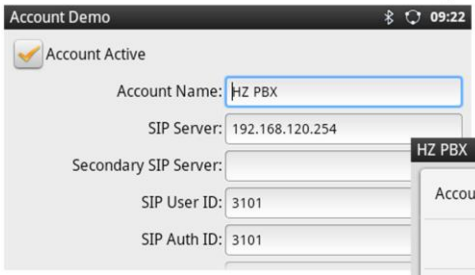
The SMS demo here shows a SIP messaging window

Click on “Enable Edit” will invoke GXP2200 built-in messaging window as shown on bottom right

Click on “Enable Message Receiver”, the demo app will receive a notification when new message arrives

Demo Android App (4)

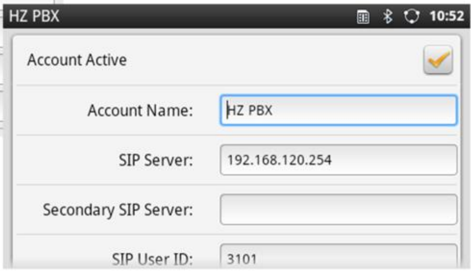
Account Demo: use API to write your own Account app



Account Demo 09:22

✓ Account Active

Account Name:	HZ PBX
SIP Server:	192.168.120.254
Secondary SIP Server:	
SIP User ID:	3101
SIP Auth ID:	3101



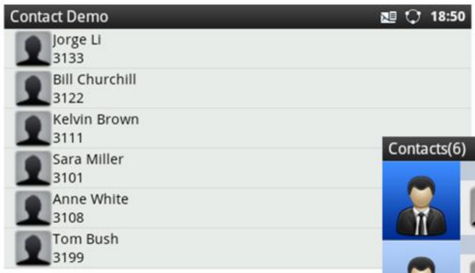
HZ PBX 10:52

Account Active ✓

Account Name:	HZ PBX
SIP Server:	192.168.120.254
Secondary SIP Server:	
SIP User ID:	3101

Demo Android App (6)

Contact Demo: use API to write your own Contact app



Contact Demo 18:50

Jorge Li
3133

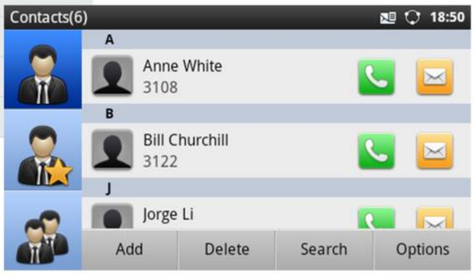
Bill Churchill
3122

Kelvin Brown
3111

Sara Miller
3101

Anne White
3108

Tom Bush
3199



Contacts(6) 18:50

A

Anne White
3108

B

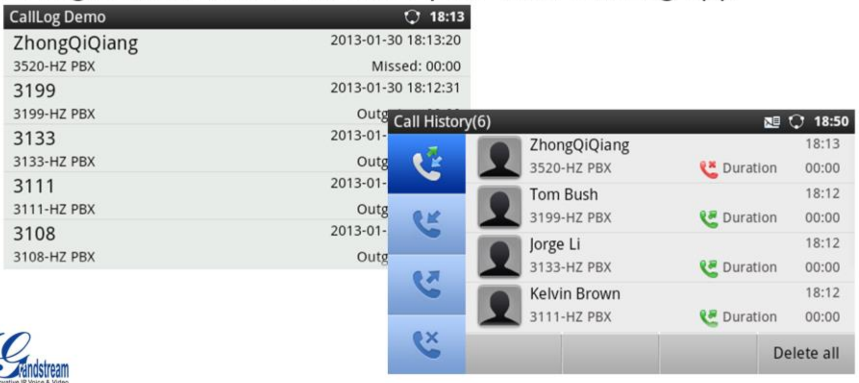
Bill Churchill
3122

J

Jorge Li

Demo Android App (7)

Call Log Demo: use API to write your own Call Log app



CallLog Demo 18:13

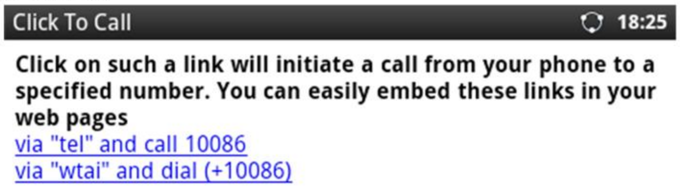
ZhongQiQiang	2013-01-30 18:13:20
3520-HZ PBX	Missed: 00:00
3199	2013-01-30 18:12:31
3199-HZ PBX	Outg
3133	2013-01-
3133-HZ PBX	Outg
3111	2013-01-
3111-HZ PBX	Outg
3108	2013-01-
3108-HZ PBX	Outg

Call History(6) 18:50

ZhongQiQiang		18:13
3520-HZ PBX	Duration	00:00
Tom Bush		18:12
3199-HZ PBX	Duration	00:00
Jorge Li		18:12
3133-HZ PBX	Duration	00:00
Kelvin Brown		18:12
3111-HZ PBX	Duration	00:00

Demo Android App (5)

Click to Dial Demo


Click To Call 18:25

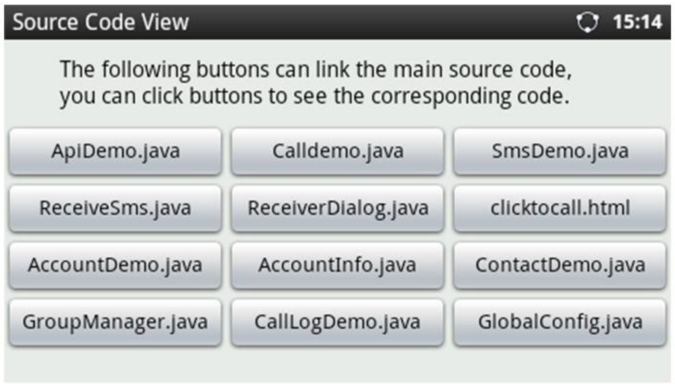
Click on such a link will initiate a call from your phone to a specified number. You can easily embed these links in your web pages

via "tel" and call 10086

via "wtai" and dial (+10086)

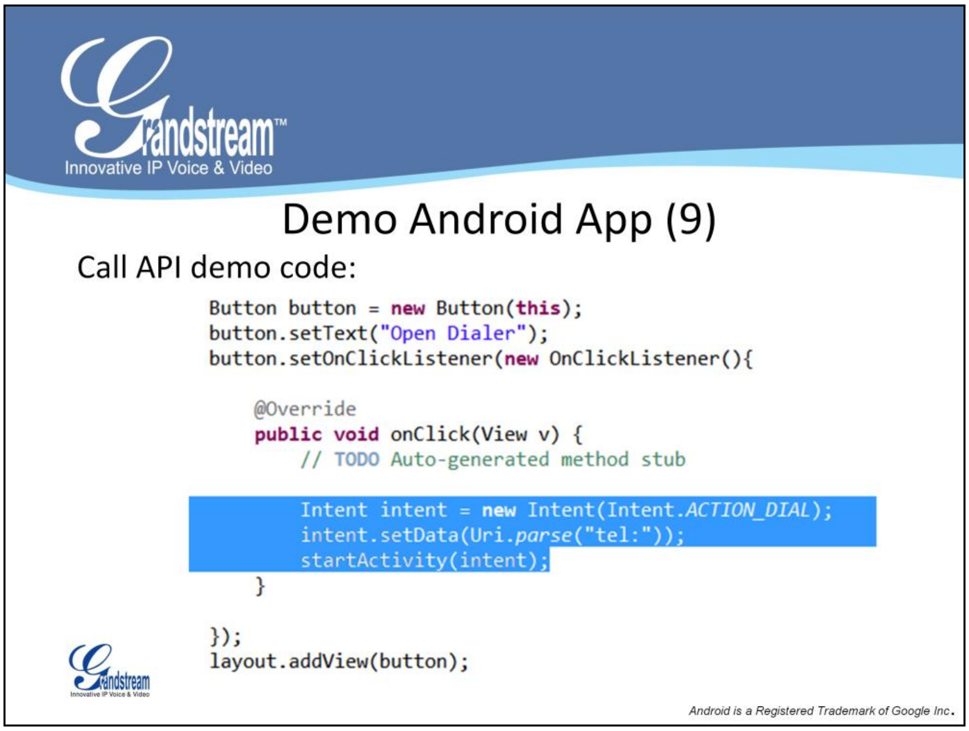
Demo Android App (8)

Source Code



Source Code View 15:14

The following buttons can link the main source code, you can click buttons to see the corresponding code.



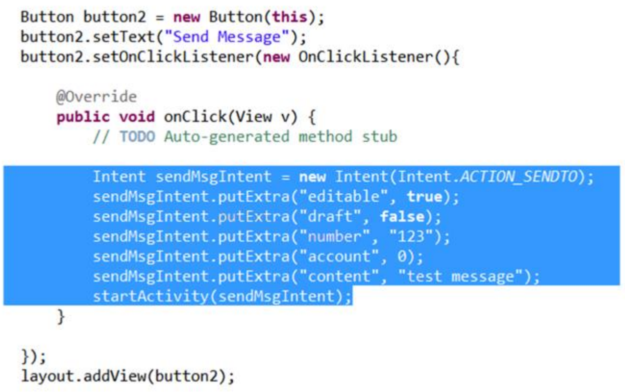
Demo Android App (9)

Call API demo code:

```
Button button = new Button(this);
button.setText("Open Dialer");
button.setOnClickListener(new OnClickListener(){

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub

        Intent intent = new Intent(Intent.ACTION_DIAL);
        intent.setData(Uri.parse("tel:"));
        startActivity(intent);
    }

});
layout.addView(button);
```

Demo Android App (10)

Send Message demo code:



```
Button button2 = new Button(this);
button2.setText("Send Message");
button2.setOnClickListener(new OnClickListener(){

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub

        Intent sendMsgIntent = new Intent(Intent.ACTION_SENDTO);
        sendMsgIntent.putExtra("editable", true);
        sendMsgIntent.putExtra("draft", false);
        sendMsgIntent.putExtra("number", "123");
        sendMsgIntent.putExtra("account", 0);
        sendMsgIntent.putExtra("content", "test message");
        startActivity(sendMsgIntent);
    }

});
layout.addView(button2);
```

Demo Android App (11)

Click To Dial demo code

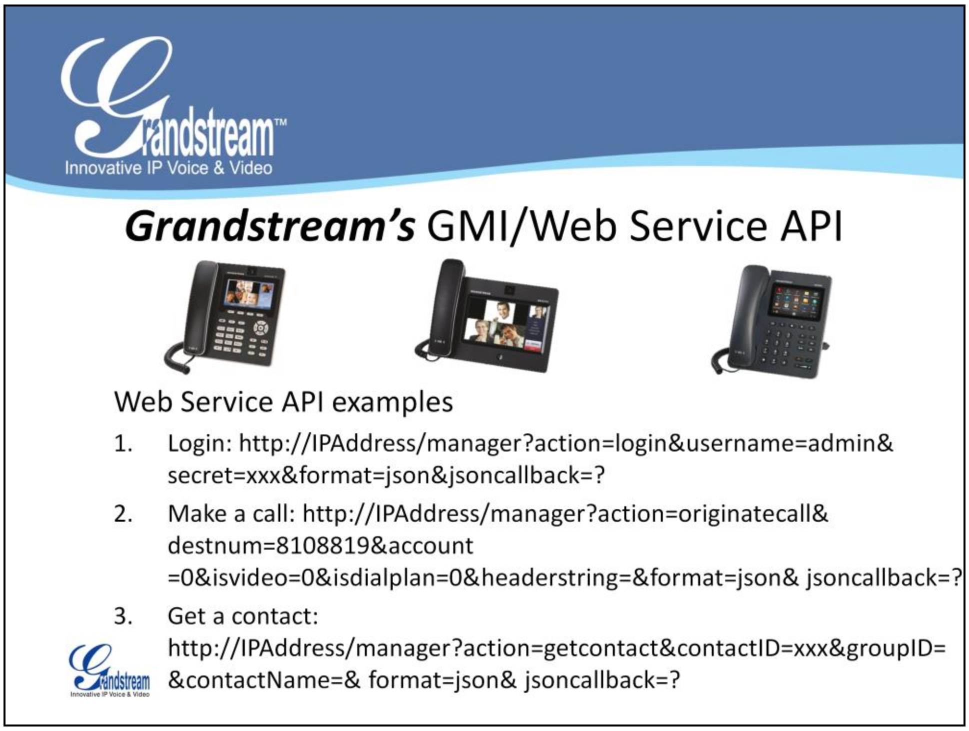
```
<html>
<body>
<b>Click on such a link will initiate a call from your phone to a specified
number. You can easily embed these links in your web pages</b><br>
<a href="tel:10086">via "tel" and call 10086</a><br>
<a href="wtai://wp/mc;+10086">via "wtai" and dial (+10086)</a><br>
</body>
</html>
```

GXP2200 APIs & Customization

1. Android API

2. GMI/Web Service API

3. GUI Customization



Grandstream's GMI/Web Service API

Web Service API examples

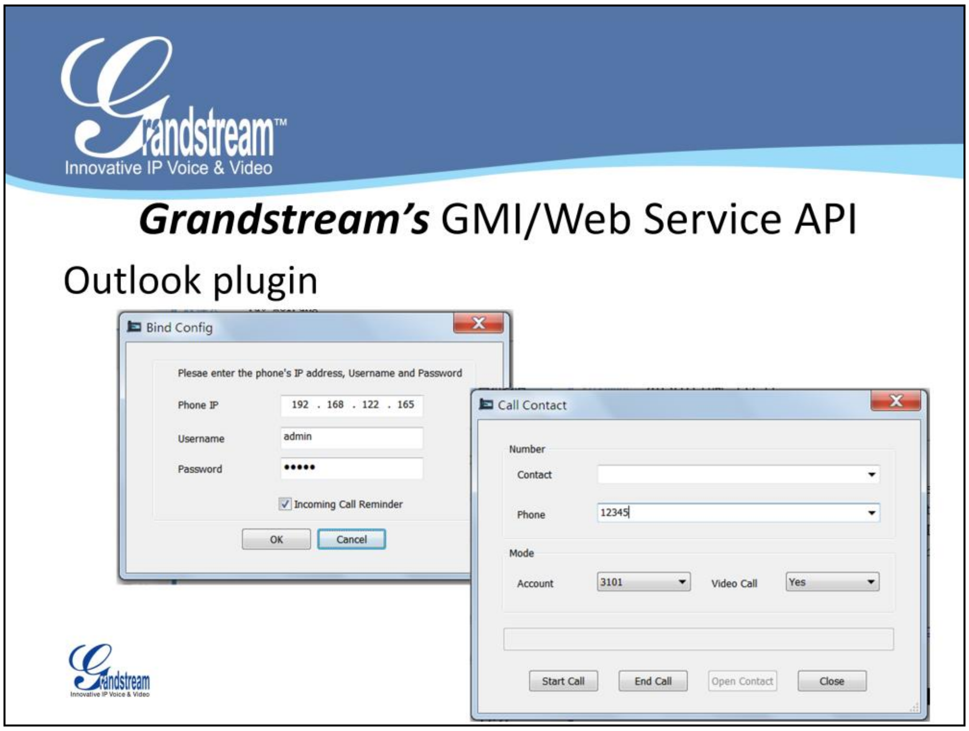
1. Login: http://IPAddress/manager?action=login&username=admin&secret=xxx&format=json&jsoncallback=?
2. Make a call: http://IPAddress/manager?action=originatecall&destnum=8108819&account=0&isvideo=0&isdialplan=0&headerstring=&format=json& jsoncallback=?
3. Get a contact: http://IPAddress/manager?action=getcontact&contactID=xxx&groupID=&contactName=& format=json& jsoncallback=?

GMI is a standard HTTP and JavaScript based, cross platform API

GMI is currently available on GXP2200, GXV3140, GXV3175 and future Grandstream products

GMI can be used to initiate calls, receive calls, get phone status, query a contact etc

Developers can develop GMI apps that use HTTP and Javascript to interact with backend servers



Grandstream's GMI/Web Service API

Outlook plugin

Bind Config

Plesae enter the phone's IP address, Username and Password

Phone IP	192 . 168 . 122 . 165
Username	admin
Password	●●●●●

☑ Incoming Call Reminder

Call Contact

Number

Contact	
Phone	12345

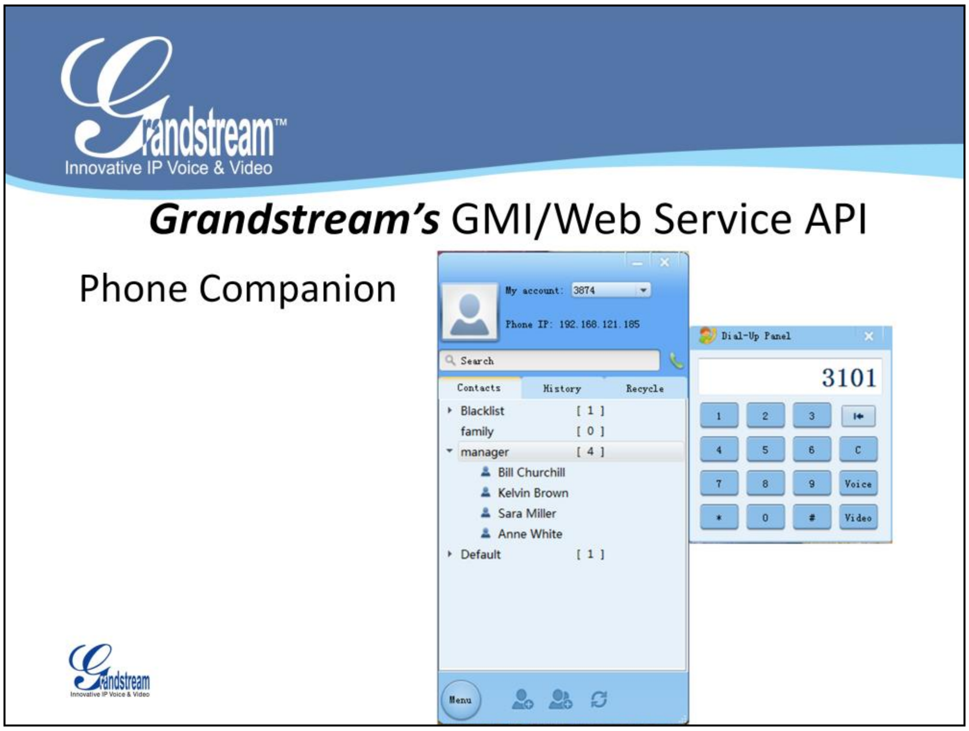
Mode

Account	3101	Video Call	Yes

The following shows several apps developed by Grandstream that are based on GMI

The outlook plugin, originally written for GXV3175 based on GMI, is also available for GXP2200

It allows outlook users to initiate phone calls directly from Outlook contact



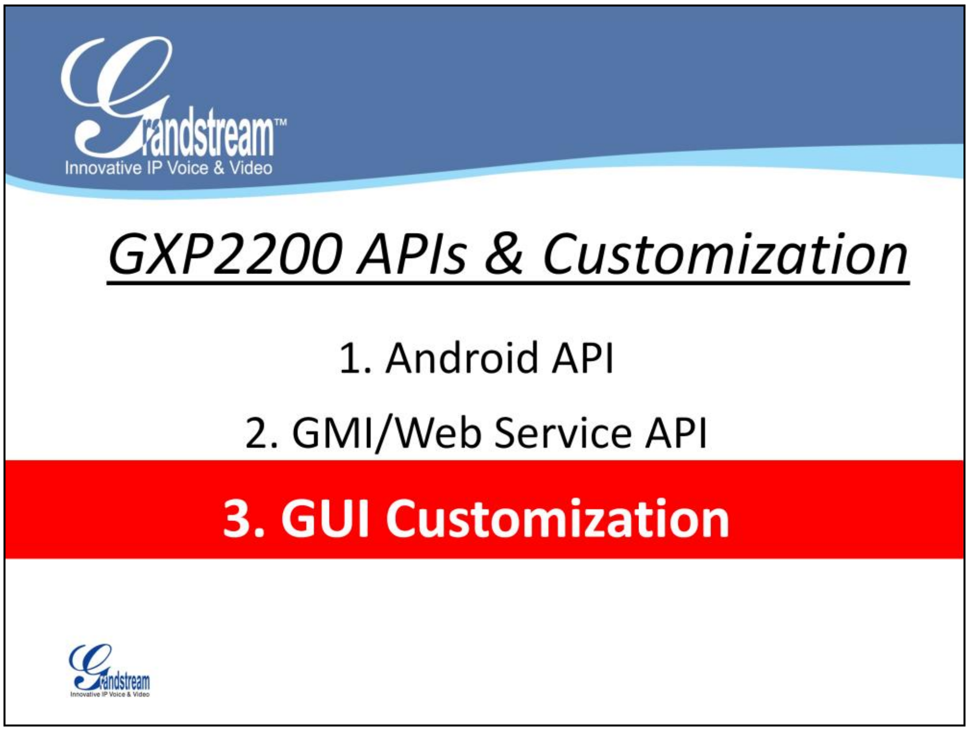
Grandstream's GMI/Web Service API

Phone Companion

The Phone Companion is a Windows program to bind the windows PC with a Grandstream phone.

Users can edit device phone book, making/receiving phone calls using this Windows program

Again it is based on GMI



Grandstream
Innovative IP Voice & Video

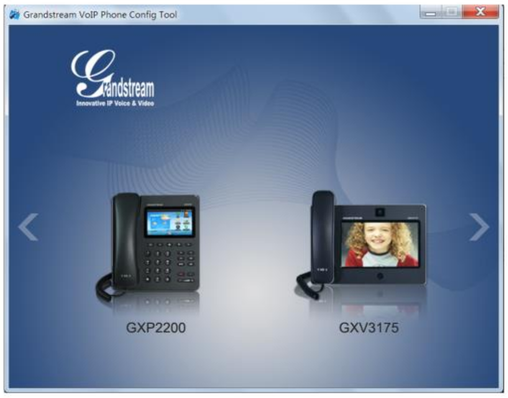
GXP2200 APIs & Customization

1. Android API
2. GMI/Web Service API
3. **GUI Customization**

Grandstream
Innovative IP Voice & Video

GUI Customization (1)

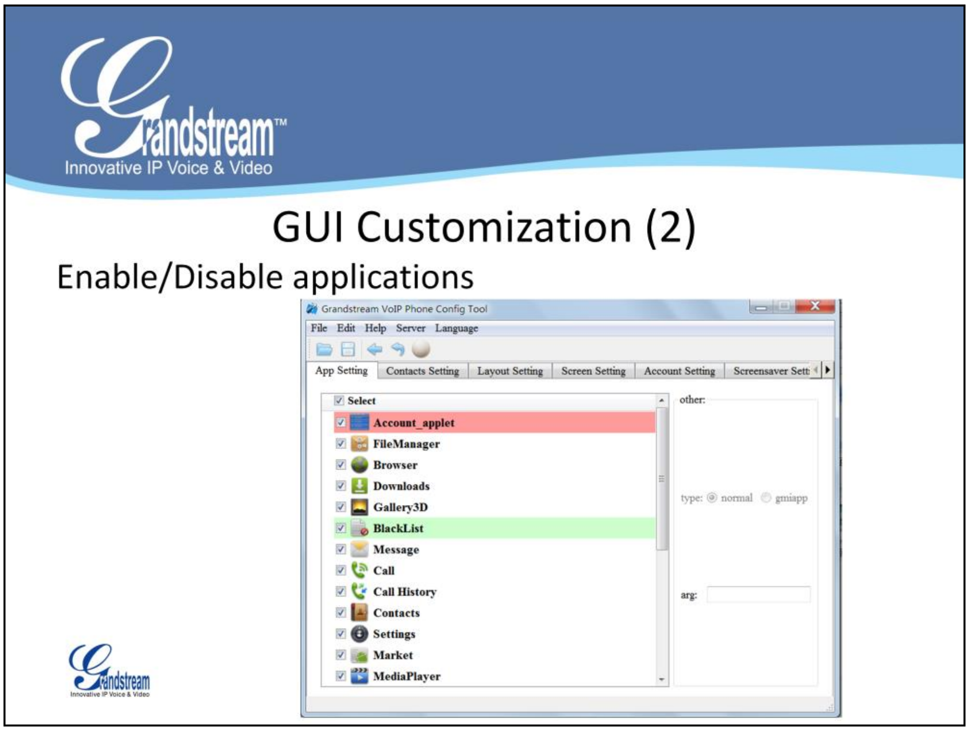
Use config tool to customize your phone



GUI customization is not really an app per se

It is a tool that runs on Windows

It can be used to customize the look & feel of the user interface of Grandstream phones, ie GXP2200, GXV3140 and GXV3175



GUI Customization (2)

Enable/Disable applications

Grandstream VoIP Phone Config Tool

File Edit Help Server Language

App Setting | Contacts Setting | Layout Setting | Screen Setting | Account Setting | Screensaver Sett

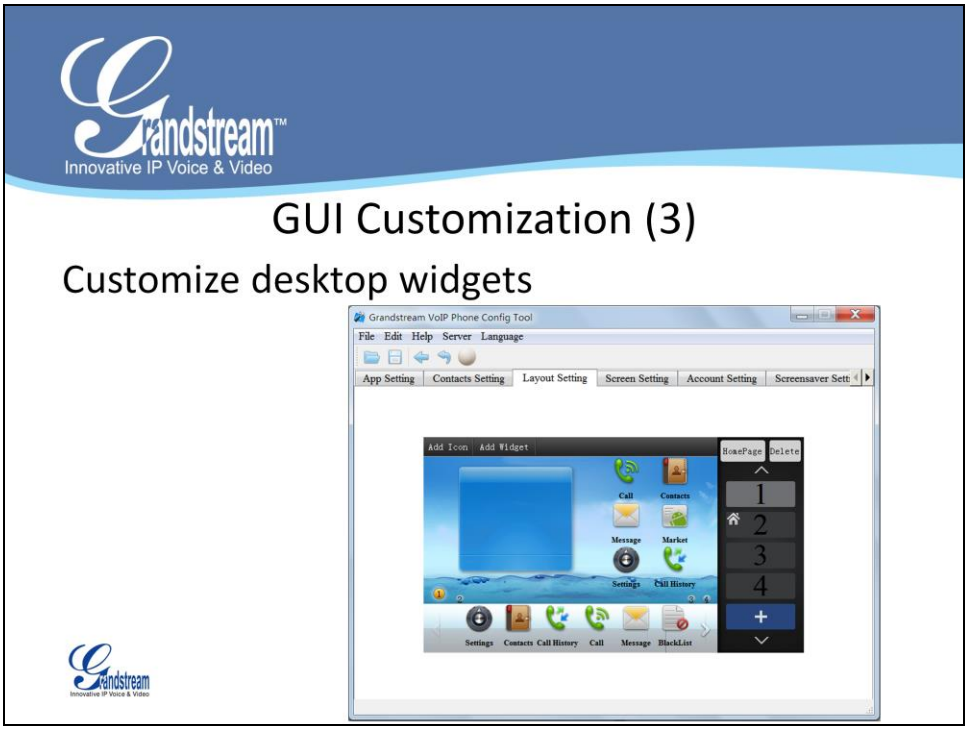
- [✓] Select
- [✓] Account_applet
- [✓] FileManager
- [✓] Browser
- [✓] Downloads
- [✓] Gallery3D
- [✓] BlackList
- [✓] Message
- [✓] Call
- [✓] Call History
- [✓] Contacts
- [✓] Settings
- [✓] Market
- [✓] MediaPlayer

other:

type: [✓ normal] gmiapp

arg:

Users can decide whether to show or hide applications

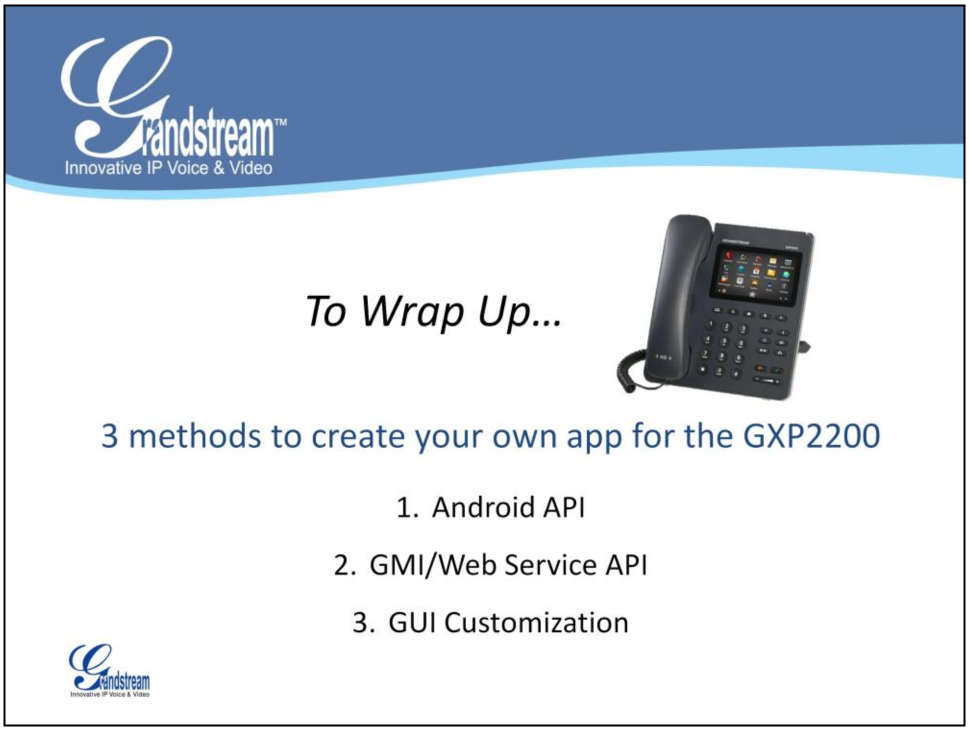


GUI Customization (3)

Customize desktop widgets

Customize what should be put on the desktop

Customize the icons, background images, screen savers etc.



Innovative IP Voice & Video

To Wrap Up…

3 methods to create your own app for the GXP2200

1. Android API
2. GMI/Web Service API
3. GUI Customization

Innovative IP Voice & Video

Grandstream™
Innovative IP Voice & Video

Grandstream
Innovative IP Voice & Video

Thank You

Android is a Registered Trademark of Google Inc.