GlassUp Developer Guide

Android

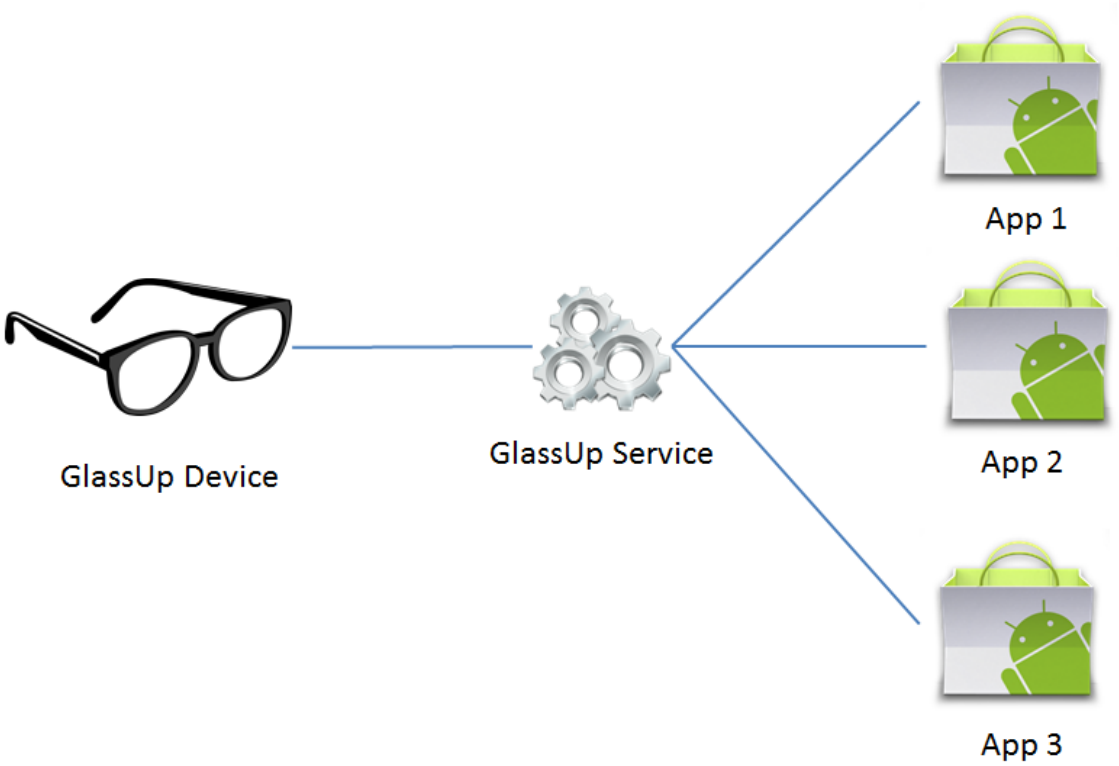
MARCH, 2015

CONTENTS

GLASSUP ARCHITECTURE

The GlassUp system is made up of:

- **GlassUp Device:** The GlassUp Device displays contents generated by the smartphone on the glasses lens.
- **GlassUp Service:** The GlassUp service is an application running on a smartphone with Android or iOS (or Windows phone). The service is the bridge between the GlassUp Device and the GlassUp Applications.
- **GlassUp Applications:** On the smartphone there are several applications providing contents to be displayed on the glasses.



GLASSUP DEVICE

The GlassUp Device is a device that display contents provided by a smartphone and send sensors data and user input to the smartphone. The communication between GlassUp Device and the smartphone is done via Bluetooth Low Energy. Inside smartphone there is an application, GlassUp Service, that is the bridge between the user application and the GlassUp Device.

The GlassUp Device has the hardware sensors and the buttons generating the notifications dispatched by GlassUp Service to the GlassUp Apps.

GLASSUP SERVICE

The GlassUp Service is the only application with a direct connection with the GlassUp Device. The GlassUp Service manages:

- Bluetooth pairing with GlassUp Device
- Device authentication
- Content upload from GlassUp Apps to GlassUp Device
- Font and Charset management
- Sensors events from GlassUp Device to Apps
- Input notification from GlassUp Device to Apps

Inside the smartphone, the GlassUp Service provides the API to upload content to GlassUp and retrieve sensors and user input notifications.

The GlassUp Service is a smartphone application that can be downloaded from the GlassUp web site. The application will be also downloadable from the Android and iOS market as soon as the GlassUp Device will be ready for the market.

GLASSUP APPS

The GlassUp Apps are the software generating the contents for GlassUp Device.

To the other direction, the GlassUp Apps receive notifications and events from the GlassUp Service and send contents to the GlassUp Service.

GlassUp will release some **Official Applications** such as email notification and translating. Moreover, with the SDK, the community can develop additional **Community Applications**.

Both official and community application must be compliant with the GlassUp API to interoperate with GlassUp Service and with the GlassUp Device.

CONTENTS MANAGEMENT

The contents generated by GlassUp applications are sent to GlassUp Service. The GlassUp Service manages the notification queue and the user interaction for the content to be displayed on GlassUp Device.

The service keeps a queue of contents to be displayed. The queue is managed as a FIFO queue with 3 priority levels:

- 1 – *alarm priority*: used by the GlassUp system
- 2 - *input priority*: used to communicate an event/response from user
- 3 - *app priority*: normal content provided by apps

The priority is not handle by Apps but by GlassUp Service. Contents generated by apps are sent with *app priority*. If the user uses a button while a content is displayed, the new content generated by the application that generated the first content has *input priority*. *Alarm priority* is used only by GlassUp Service contents as low battery, system disconnected, etc.

The user input is dispatched to the app that created the content. If the foreground App respond to the notification with a new content, the last one has higher priority on the queue and is displayed as soon as possible. This behaviour allows to display more consecutive contents from the same app.

Each content is displayed for no more than *maxPersistenceTime*. After the visualization, the content is removed from the queue. A notification is sent to the app to notify the content displayed.

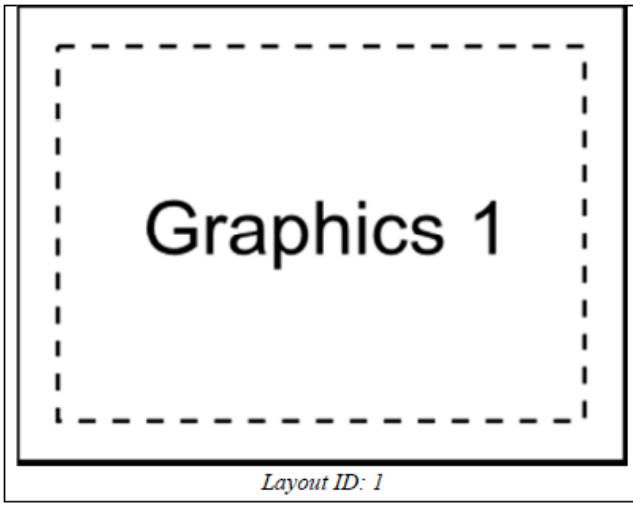
The content on the GlassUp Device is provided by an application at time.

CONTENT CREATION

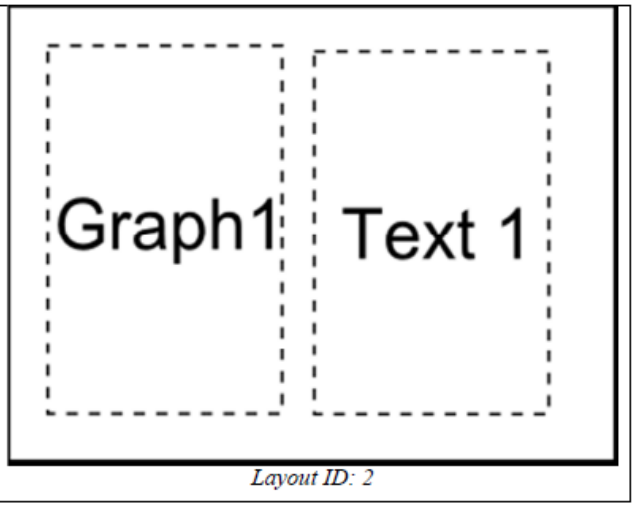
A content is a visual message displayed on GlassUp Device. To create a content a GlassUp Application has to select a Content Layout and fill it with graphics and texts.

GlassUp Service provide some predefined layouts.

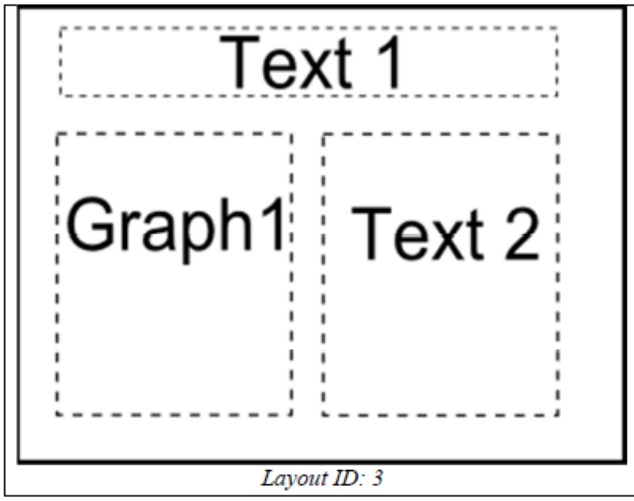
Layout ID 0 is provided to display only Text, other layout Type are following:



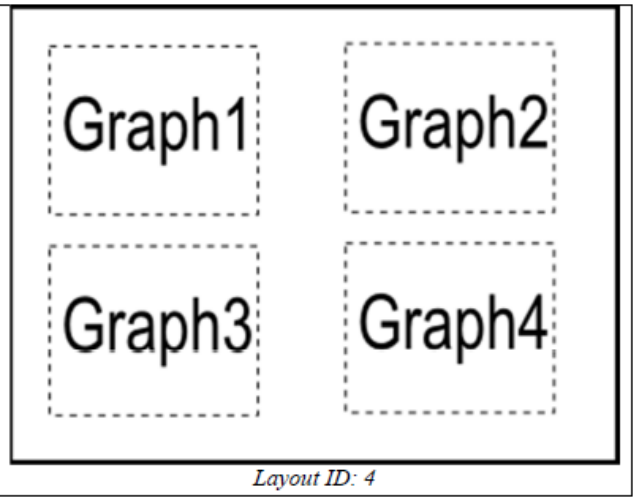
Layout ID: 1



Layout ID: 2



Layout ID: 3



Layout ID: 4

GRAPHICS

Graphic must be pre-uploaded to the GlassUp Device to have it available. During the first connection between a GlassUp App and GlassUp Service the App provides the graphics that will be pre-uploaded to to GlassUp Device. After the graphics are uploaded the GlassUp App can create a content selecting a layout and the associated graphics ID.

Graphics contents are displayed inside a graphics area of a layout. The graphic is cropped to the layout graphic area and is rendered starting from the top left corner.

GlassUp Application must use GlassUp Graphic content. A tool, GlassUp Image Converter, help developers to convert graphics and Images to the GlassUp Graphics format. The graphic content in the GlassUp Device are monochromatic and with 1 bit of colour depth.

TEXT

The font and the charset are selected by the user and are managed globally. Every text used inside layouts are rendered using the selected font.

User selects font and charset in the GlassUp Service Settings.

The maximum text available on a screen is of **6 row** with **12 character for row**.

INPUT FROM USER HANDLING

The user inputs are sent to the app providing the content displayed: the *foreground* app.

If no content is displayed there is no foreground app. In this case, the user input is sent to a default application configured by the user.

Some inputs are managed by the GlassUp Device itself to display the GlassUp Device status and to activate the configuration mode.

INPUTS

GlassUp has two buttons called button A and button B.

Each button emits the clicked notification when it is activated by the user. Buttons also provide less commonly used notification as pressed and released and longPressed. The notification sequence is:

 pressed - long pressed - long pressed - long pressed - ... – released

GLASSUP SETTINGS

 Font: select font e charset

 maxPersistenceTim*e*: maximum visualization time “on lens”

 minPersistenceTime: minimum visualization time “on lens”,

 default application: app receiving user input if there is no app on foreground

GLASSUP API REFERENCE

IN THIS SECTION A GENERAL HIGH LEVEL PRESENTATION OF THE API IS PROVIDED. A MORE DETAILED DESCRIPTION IS PROVIDED IN THE "SDK" SECTION WITH API REFERENCE FOR ANDROID, IOS AND WINDOWS.

CONNECTION API

This API provides information about the GlassUp status and properties.

- int getConnectionStatus(): return the connection status with the GlassUp Device:
  - 0: UNCONNECTED
  - 1: CONNECTING
  - 2: CONNECTED

A notification is sent to all connection listeners when the connection status changes. All connection status is in GlassUpServiceInterface.

DRAWING API

The basic API to draw content on the GlassUp is:

- GlassUpAgent.sendContent(contentID, layoutiID, graphicID[],texts[]): This function sends a content composed of layout, graphics and texts to the GlassUp Service. The GlassUp Device will display the content as soon as the content will be received. The contentID must be generated by App and can be any int number.
- GlassUpAgent.sendConfiguration(drawable) : This function preload graphics to the GlassUp Device and map the graphics ID to be used in the sendContent function. Graphics ID of images corresponds to bitmap Array index that you pass to sendConfiguration method.

GRAPHICS

The GlassUp Device can display monochromatic contents with 1 bit level of colour. The contents has 320x240 pixel size. The top bar, 320x24 pixel, is managed by GlassUp Service to display status information as battery level and connection status.

DISPLAY

The display available for applications has 320x216 pixel size.

COORDINATE SYSTEM

The coordinate system to draw on the lens begins in the upper left.

The **X** axis is horizontal and increases to the right side.

The **Y** axis is vertical and increases to the bottom side.

X and Y offset and sizes are represented by int.

DRAWING PROCESS

The drawing process lives inside the GlassUp Apps. The Apps build a content that will be displayed on the GlassUp Device mixing a layout with graphics and texts.

First time your start your app or every time you change image content of app you have to call *sendConfiguration(drawable)* through your agent.

GlassUp Apps send the content to the GlassUp Service with a contentID, then service queues the contents and call onContentResult on all ContentResult Listeners.

When the content is sent to GlassUp Device an onContentResult is fired with the contentID parameter and status of content sent. Status are mapped on GlassUpServiceInterface. If a disconnection with GlassUp Device occurs, every contents are dropped and sender app receives status GlassUpServiceInterface.*STATUS_ERROR.*

SENSORS API

On GlassUp Device, there are an accelerometer, a compass and a light sensor. There are also a battery charge sensor and two input buttons.

GlassUp Service manages the communication with the sensors on the GlassUp device and provides access to those sensors to the Apps.

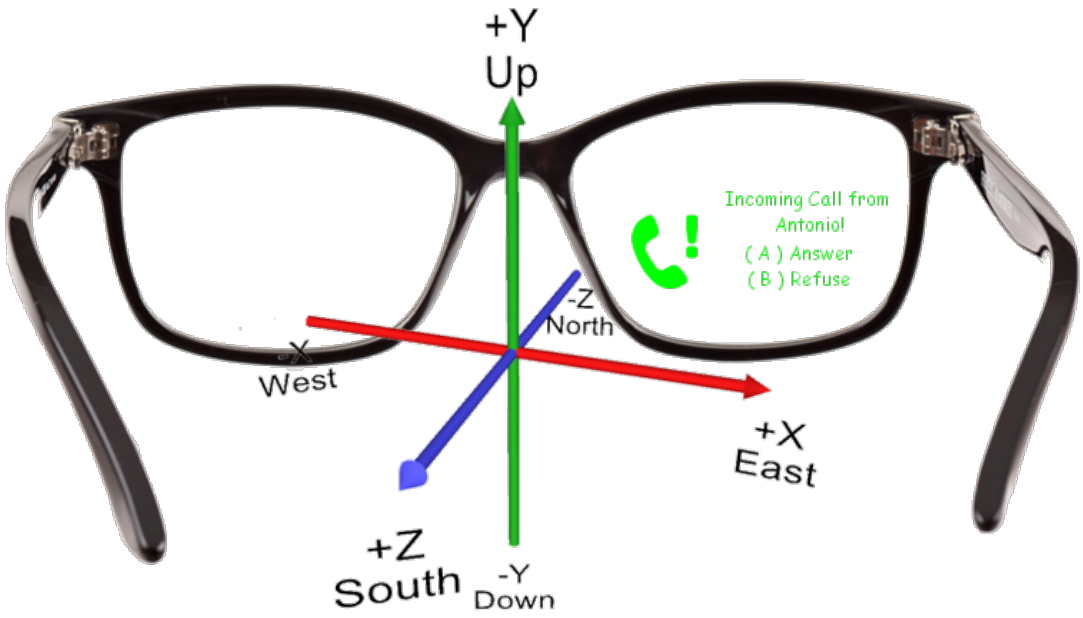
A notification event is generated when a sensor change its value with a maximum frequency of 1Hz.

COORDINATE SYSTEM

The coordinate system used on some sensors notification is drawn on the following image.

Wearing GlassUp:

- X-axis: is parallel with the lens frame. The direction is toward the user right
- Y-axis: is oriented from up to down
- Z-axis: lay on the glass temple and is oriented toward the back



EVENT NOTIFICATION

Periodically sensors update their values and an event is sent to the Apps. The event provides the event type and the event values. The GlassupSensorEvent provides the sensor type and an array of 4 float for the values. The following sensor sections describe how to decode the values on the event.

To receive event on your app you must subscribe to an event via registerToEvent(type) method and unregister via unregisterToEvent(type), type must be an int specified in class GlassUpEvent.

- **Battery Level**: The battery level sensor provides the battery charging level.

Value index	Description
0	battery charging level. From 0% to 100%
1	Flags: • 0x01 : Battery on charging • 0x10 : Battery broken
2	not used
3	not used

- **Buttons Inputs:** On the GlassUp there are two touch buttons: A and B. Button A is the button closest to the lens.

Value index	Description
0	button generating the event: • 0 : A (plus) • 1 : B (minus)
1	button event type: • 0 : pressed • 1 : released • 2: click • 3 : long press
2	not used
3	not used

- **Accelerometer**: The accelerometer sensor notifies the acceleration of the GlassUp. The acceleration has the gravity included.

Value index	Description
0	Acceleration on X axis
1	Acceleration on Y axis
2	Acceleration on Z axis
3	not used

- **Compass:** The compass returns the orientation angle from the north pole (Yaw angle) but also the device orientation.

Value index	Description
0	Yaw angle
1	Pitch angle
2	Roll Angle
3	not used

- **Ambient Light Sensor:** The light sensor can sense light from XXX to XXX lux.

Value index	Description
0	light level. From 0% to 100%
1	not used
2	not used
3	not used

ARCHITECTURE

IN THIS SECTION WILL BE SHOWN THE CONNECTION BETWEEN THE EMULATOR AND THE DEVICE THAT USES THE GLASSUP API. FINALLY WILL BE SHOWN HOW TO INTEGRATE THE SDK INTO OWN ANDROID PROJECT.

LOCAL NETWORK CONNECTIONS

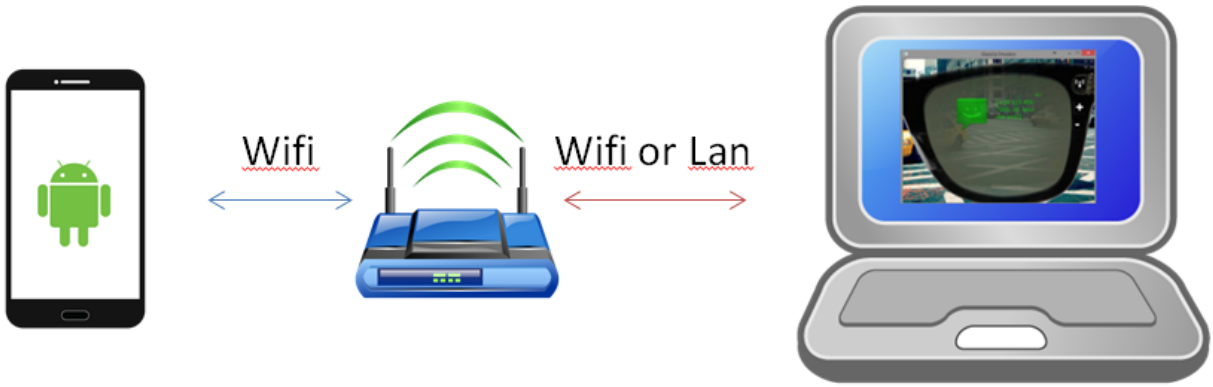


Figura 1 Network Connections

ANDROID SERVICE SETUP

To install the GlassUp service on own Android device you must use the adb(Android Debug Bridge) tool. You can find it into Android SDK platform-tool folder. After that, you must install the GlassUpService.apk, you can find it into the root of the attached zip file. The step to install the app on own device is the same for Windows and MacOSX.

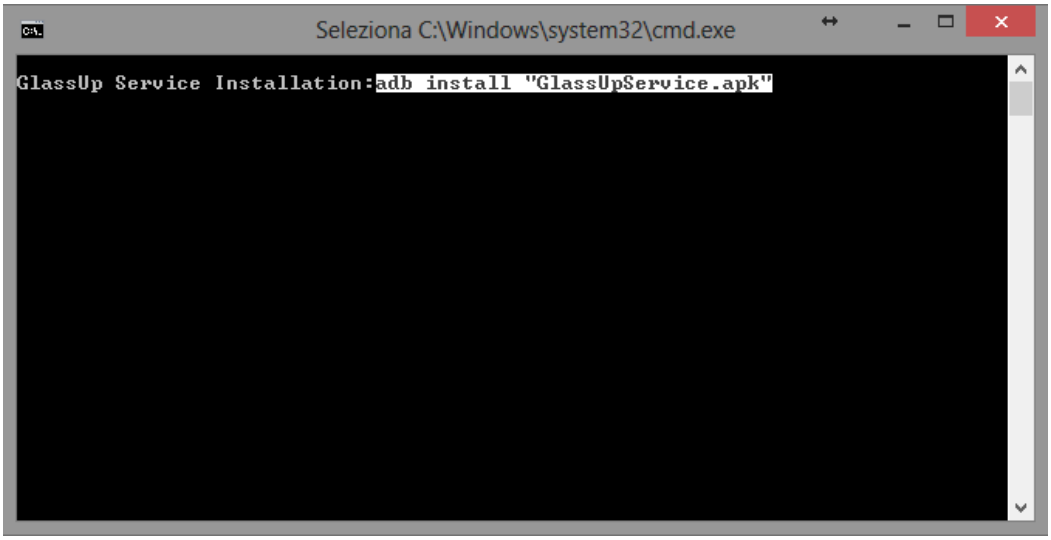


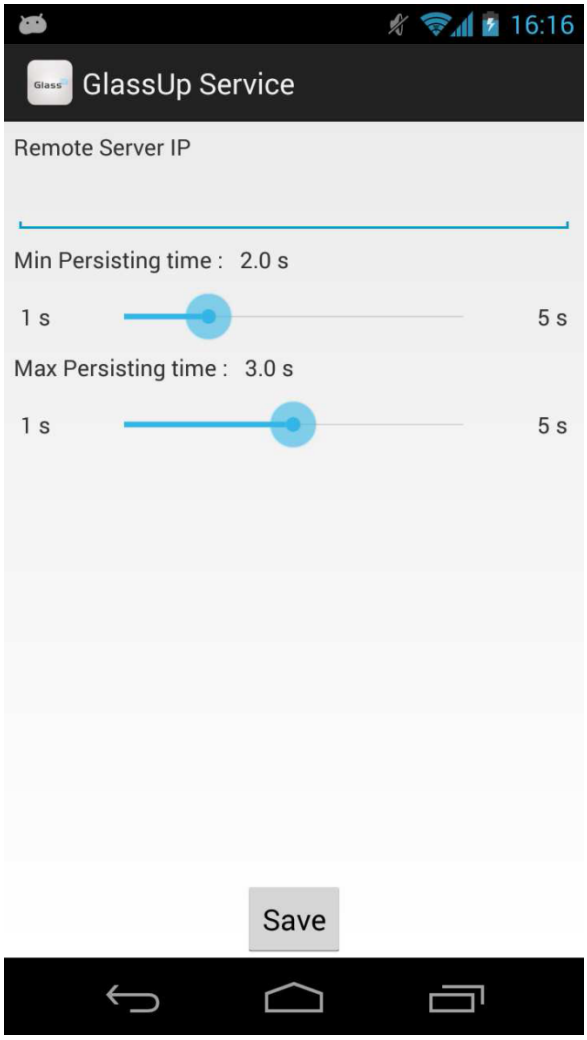
Figura 2 GlassUp Service installation

SERVICE CONFIGURATION

Ensure that your GlassUp Emulator is opened and in the same Network of the Android device.

Then on your Android Device Start the App: GlassUp Service.

After start, you can see following screen:


16:16
GlassUp Service
Remote Server IP
Min Persisting time : 2.0 s
1 s
5 s
Max Persisting time : 3.0 s
1 s
5 s
Save

Then type your GlassUp Emulator IP (Your machine running emulator IP address) in the first field then click save and you will see the app close and a Notification icon appears on the Notification Bar.

Now your Service is started and is also connected to the GlassUp Emulator.

If you see “Connection Status: unconnected” ensure that your GlassUp Emulator is started properly and your Network is the same of your Computer running Emulator.

EMULATOR SETUP

The emulator is available for Windows and MacOSX. You can find both the setup into the attached zip file. When you start the emulator you must disable the firewall or control if the port 2000 is not blocked.

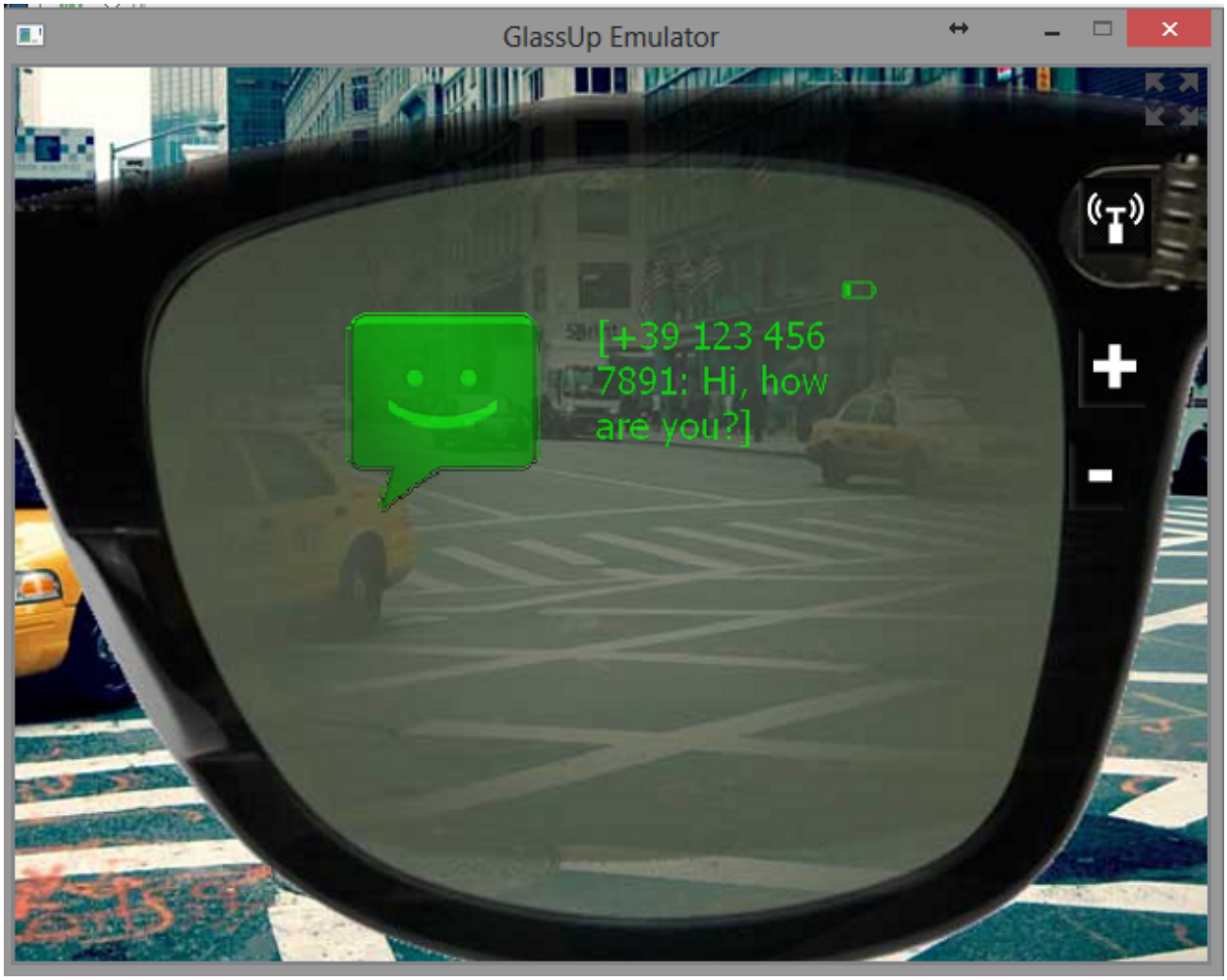


Figura 3 GlassUp Emulator on Windows 8

ANDROID API REFERENCE

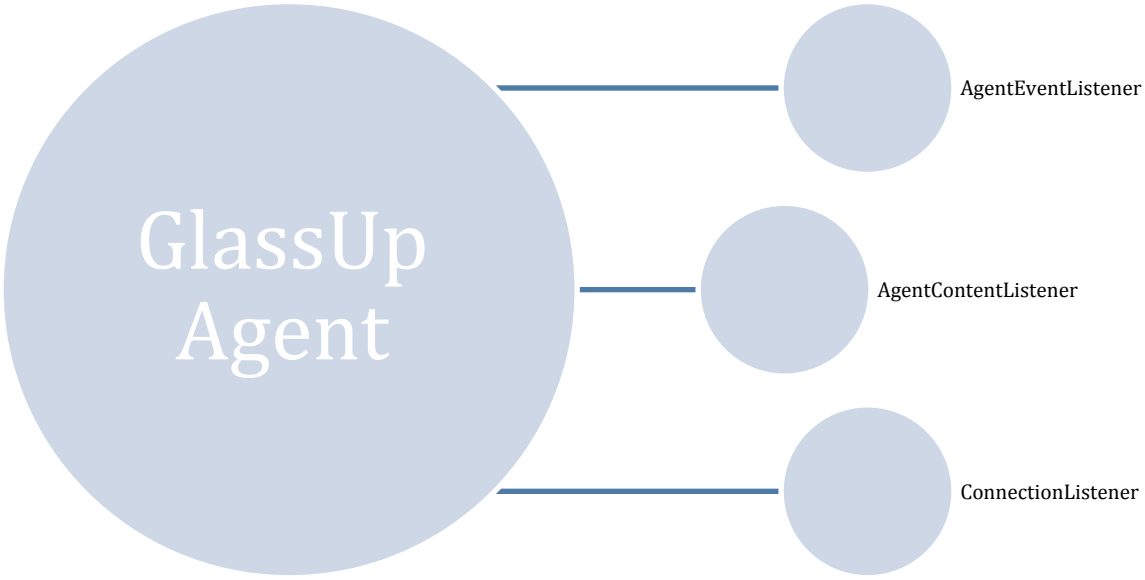


Figura 4 Agent Architecture

The GlassUpAgent is a class that connect you app to the GlassUpService. With this class you can send a content to device or get information with its Listener interface.

The ConnectionListener receives the state of connection, the ContentResultListener receives the content results and EventListener receives the events callbacks. All of this listener will be must set to the instance of own Agent through its method:

- setEventListener(AgentEventListener);
- setContentResultListener(AgentContentListener)
- setConnectionListener(ConnectionListener)

After that, we must send the configuration to the GlassUp device. To make this we can use 2 public functions of the Agent object:

- sendConfiguration(int[]): this method accepts an array of drawable to send to the device
- sendConfigure(Bitmap[]): this method accepts an array of Bitmap to send to the device.

If you have an image into drawable folder of you project, the sendConfiguration function is so perfect, but, if you have an image created at runtime the sendConfigure method is perfect.

REMEMBER: you must send the configuration when the app starts and when you want change the images!

At this point the connection with GlassUp Service is established and a function is provided to display contents on the glasses (if app is configured).

To send a content to the device use the sendContent() function of your Agent class instance. This function accepts the below parameters:

- contentId: is the Id of the content that you want to send. This int variable is a counter;
- layoutId: is the layoutId(0-4) to show your informations;
- String[] indexImage: is the position of the images that you want to show on the layout ;
- String[] text: is the strings that you want to show on the layout;

EXAMPLE:
Agent.sendContent(index++,1, new String[]{String.valueOf(iconPosition)}, new String[]{"HELLO!"})

Every time you send a content, service notifies you result of this command through onContentResult method of AgentContentListener with contentID parameter.

IMPORTANT:

Remember that the GlassUpAgent must follow Activity lifecycle. So you have to call appropriate methods:

- **public void** onCreate(Context context);
- **public void** onResume();
- **public void** onPause();
- **public void** onDestroy();

in your Activity corresponding methods.

EXAMPLES

IN ADDITION TO THIS DOCUMENT SOME EXAMPLES WITH EXPLANATIONS ARE INCLUDED WITHIN THE KIT. EACH FOLDER IS A SINGLE PROJECT THAT YOU CAN IMPORT IN YOUR ECLIPSE IDE

SUPPORT