
# Getting started
# Real Estate Portal IRIS
# User manual and features

**Version: 07/2013**

EUROS Embedded Systems GmbH
Campestraße 12 | 90419 Nürnberg
Fon: +49-911-300328-0 | Fax: +49-911-300328-9
Web: www.euros-embedded.com
eMail: support@euros-embedded.com

# Contents

## Getting started in four steps:


# Setting up the terminal


# Introduction of Real Estate Portal IRIS


# Working with existing digital contents


# Miscellaneous

# Installing the terminal

- A terminal consists of hardware and software components:
  - IRIS display terminal
  - Android OS version 4.0.4 with default configuration files
  - Graphical user interface application (Carousel APK)
  - Terminal control and monitoring application (Wave5CTRL)
  - Optional set of configuration files

A fully equipped terminal will automatically boot Android after being powered on and automatically run the Carousel APK and Wave5CTRL applications after operating system has been started.

Each terminal system interacts with the following software components:

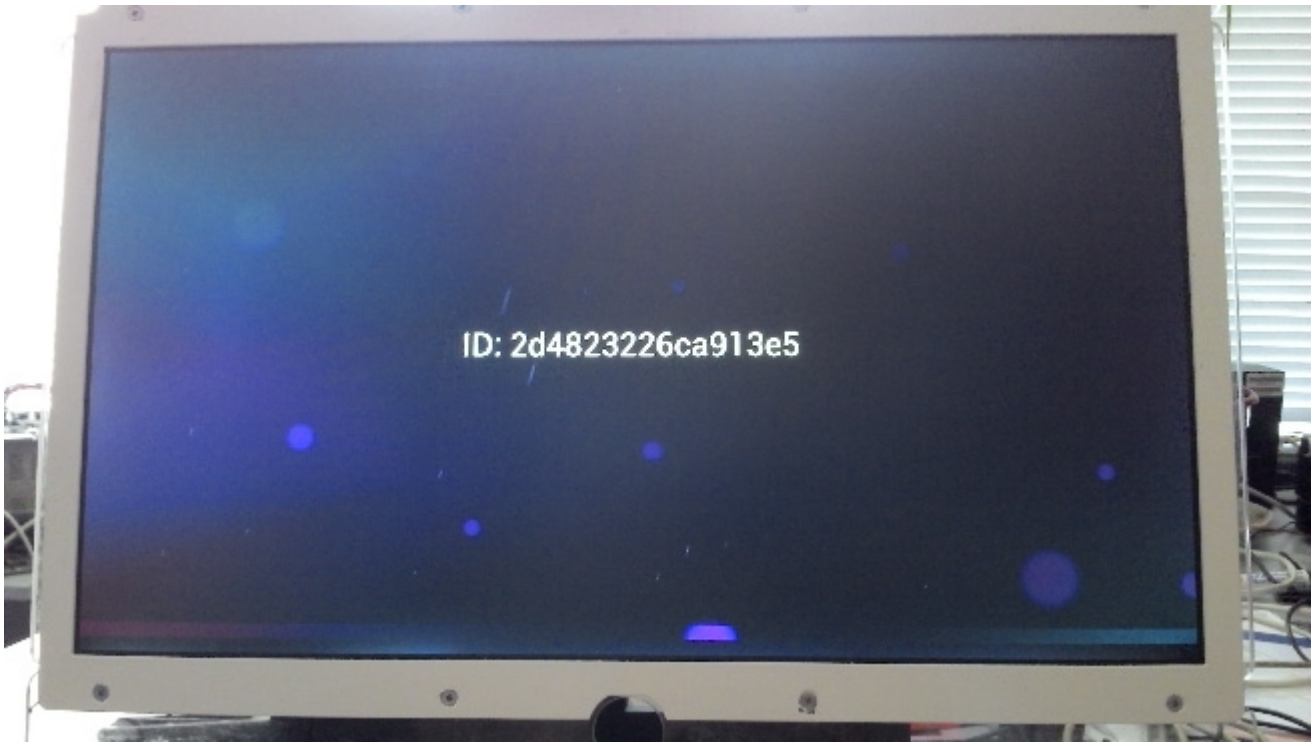
- centralized database system, storing information on all available offers
- web application offering access to this centralized database system
- web application offering user interface for creating/modifying terminal system configuration files
- mobile phone application allowing to export information collected on the phone to the centralized database
- web application offering import of OpenImmo real estate data
- GUI tool for automating configuration changes and configuration updates

# 1.1 Android OS version 4.0.4

Terminals come with preinstalled Android ICS built for Telechips microcontroller TCC8920. It is a standard Android build with the following extensions:

- Dedicated driver for brightness control (using TCC PWM module)
- Dedicated driver for camera module and v4l layer driver

Each terminal has a unique identifier assigned to it, that can be used to identify and address the terminal, modify its configuration and assign/request data for it.




Terminal identifier is shown by the Carousel GUI application when it is started (Identifier is shown only when Carousel application is started and there are no configuration files available). Once this identifier is shown, a system administrator (user) should log in with his credentials to configuration site available at:

*http://www.wave-five.com/cnfms/*

and register the terminal. After a terminal has been registered, a default configuration will be created for it and downloaded automatically to the terminal device.

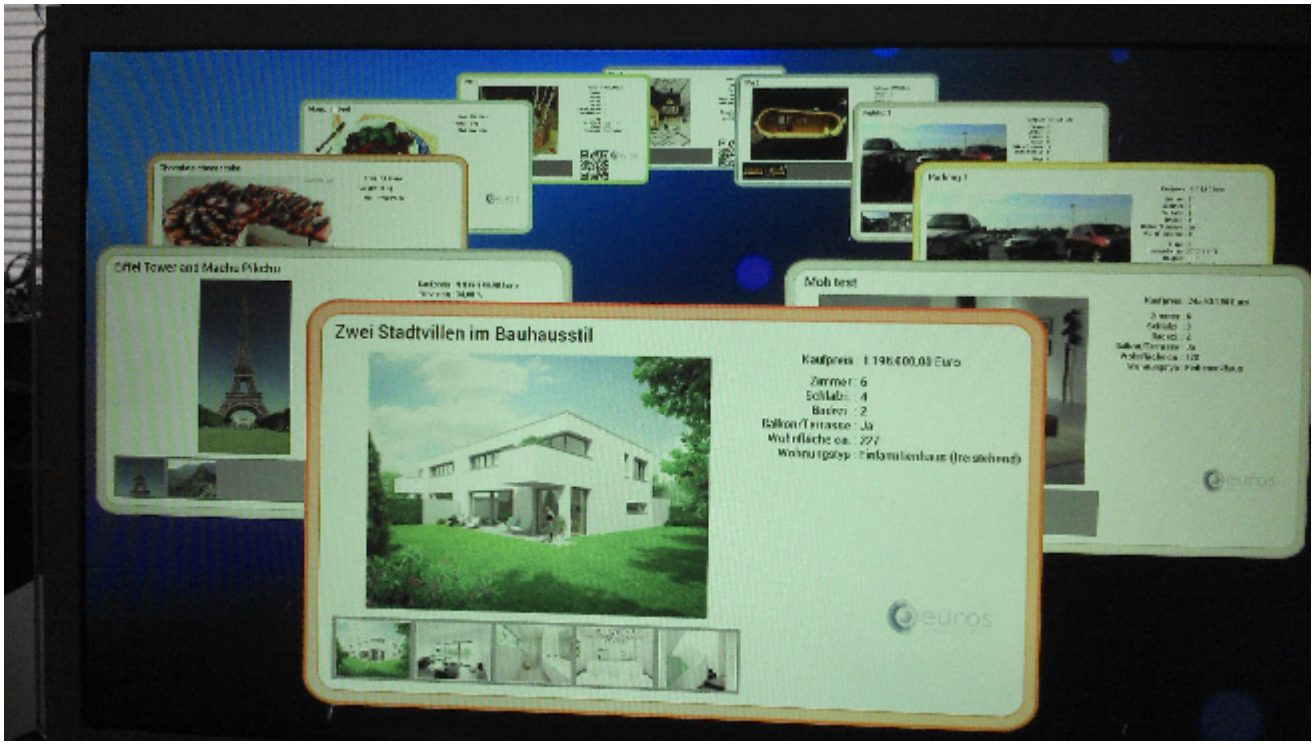
**Note: Terminals delivered with standard configuration inside will not show this message at startup. The idea of this message is to assign system administrators into registering the terminals for a first time.**

**Normal users are not expected to see it or make use of this feature.**

# 1.2 Graphical user interface application

Graphical user interface application is essential for providing terminal services[1]. It is responsible for:

- Display of available digital contents in user-friendly way (e.g. as carousel)
- Download and regular update of available information from centralized database
- Download and update of configuration files from centralized configuration database
- Automatic (or manual upon request, when there is a keyboard available) change of displayed items and navigation between their visual elements (detailed pictures)




A carousel type of graphical user interface application is started automatically upon system boot and remains active as long as the terminal is running. In case of a problem (for example software exception or power failiure) the application is automatically re-stated to ensure that there is always an active carousel application running in full-screen mode.

1. Herein “terminal services” refers to showing digital contents in a structured way and updating it regularly

# 1.3 Terminal control and monitoring application

Terminal control and monitoring application is running in the background (without accessible visual elements). Primary responsiblities of this application are:

- Continuously monitor the system temperature and in case it reaches pre-defined boundaries to try to decrease it by reducing brightness of the screen.
- Check regularly for available updates for Carousel application (and install them when available)
- Check regularly for available updates for control and monitoring application (and apply them in case they are available)
- Report temperature and usage statistics to a centralized database system
- Check if the Carousel application is running and restart it if necessary


Monitoring and control application makes use of temperature and brightness sensors to read information about system status and take corrective actions if necessary.

When temperature rises above a dangerous level (configured in a XML file) then monitoring and control application starts to reduce screen brightness (steps can be configured also in the XML file, but default step is to reduce/increase current level by 5%).

# 1.4 Default set of configuration files

Prior to its delivery to a customer a terminal system has to be configured. This section gives a short overview of the configuration files, their structure and default values.

## 1.4.1 Configuration of Carousel application

Carousel application reads the basic user configuration from XML files during its initialization. There are four XML files which determine the particular configuration and they are described below in this document. All of the files are stored on the Android device in /sdcard/demo/ directory

- **config.xml** - the main configuration file

<*DeviceName*> element which contains the name of the device. It is reserved for future use.

<*RequestInterval*> time interval in seconds for checking new/updated digital contents in the database. If this element is missing or not set properly, then the default interval will be set by the application (60 seconds).

<*ActiveFilterSet*> name of a filter from filters.xml which determines the digital contents from the database to be loaded in the carousel.

<*ActiveCommandSet*> name of a command set from commands.xml. The selected command set controls the carousel actions.

<*Path*> sets the name of the trajectory of the carousel. The application searches the particular trajectory in trajectories.xml and if it is not found the application calculates the coordinates. So far there are three types of trajectories: "horizontal", "vertical" and "oval". If the trajectory is not set, then the application accepts the default trajectory - "horizontal".

<*DeviceID*> is the identifier of the particular terminal. It is set automatically from the carousel application.

- **commands.xml** - contains command sets for control of the carousel

<*CommandSet*> every element of this type collects sufficient description of commands to determine the behavior of the carousel. It has one attribute called "name" that identifies the command set. The content of a command set might be elements which refer to particular commands. The supported commands are:

- *Repeat* (has attribute "count" that defines the number of repetitions)
- *Wait* (has attribute "value" which determines the wait time in milliseconds)

- *Enter*
- *TurnRight*
- *TurnLeft*
- *GoBack*.

*Example contents of commands.xml:*

```
<?xml version="1.0" encoding="UTF-8"?>
<Commands>
<CommandSet name="demo">
<Repeat count="inf"><Wait value="4000" />
<Enter /><Wait value="5000" />
<Repeat count="[count]-1">
<TurnRight />
<Enter /><Wait value="3000" />
</Repeat>
<GoBack /><TurnRight />
</Repeat>
</CommandSet>
<CommandSet name="demo3"><Repeat count="inf"><Wait value="2000"/><TurnLeft /></Repeat></CommandSet>
</Commands>
```

- **filters.xml** - contains filters of offers

*<FilterSet>* is the main element in filter.xml. There can be more than one element from this type. The content of FilterSet should be sub-elements of type FilterGroup.

The attribute "maxcount" stores the maximum number of digital content items which can be loaded in the carousel.

The attribute "name" identifies the particular FilterSet.

The attribute "logic" sets the logic operation which will be applied for its sub-elements in the FilterSet to filter offers. The supported logic operations are "logical and" and "logical or".

*<FilterGroup>* set of simple sub-filters. FilterGroup has one property - "logic". It determines the logic operation to be applied for its sub-elements. Operations can be "logical and" and "logical or". Sub-elements of FilterGroup are:

- *AgentUserName*
- *Area*
- *Year*
- *Balcony*
- *ContactName*
- *City*
- *Type*.

All of them have one attribute called "cond" and their content is text. The sub-elements refer to fields in the database. The attribute "cond" sets the condition in the filter. The supported conditions are: "*equals*", "*not_equal*", "*contains*", "*not_contain*", "*greater*", "*less*".

The conditions are applied for the content of a sub-element of FilterGroup versus the values for a particular field in the database.

*Example contents of filters.xml:*

```
<?xml version="1.0" encoding="UTF-8"?>
<Filters>
<FilterSet name="test" logic="and" maxcount="8">
<FilterGroup logic="and">
<AgentUserName cond="equals">admin</AgentUserName><Year cond="greater">1999</Year>
</FilterGroup>
<FilterGroup logic="or">
<City cond="not_equal">Munich</City><Year cond="less">2013</Year>
</FilterGroup>
</FilterSet>
</Filters>
```

- **trajectories.xml** - description of the possible trajectories of the carousel.

If this XML file does not exist, it can't be read or its data is corrupted then the application calculates each coordinate of the selected trajectory.

<*Path*> element which describes one trajectory. It has two attributes - "name" and "width". There can be a third attribute called "default", but it is reserved for future use.

The attribute "name" identifies the trajectory.

The attribute "width" shows the width of the display in pixels for which the trajectory is calculated.

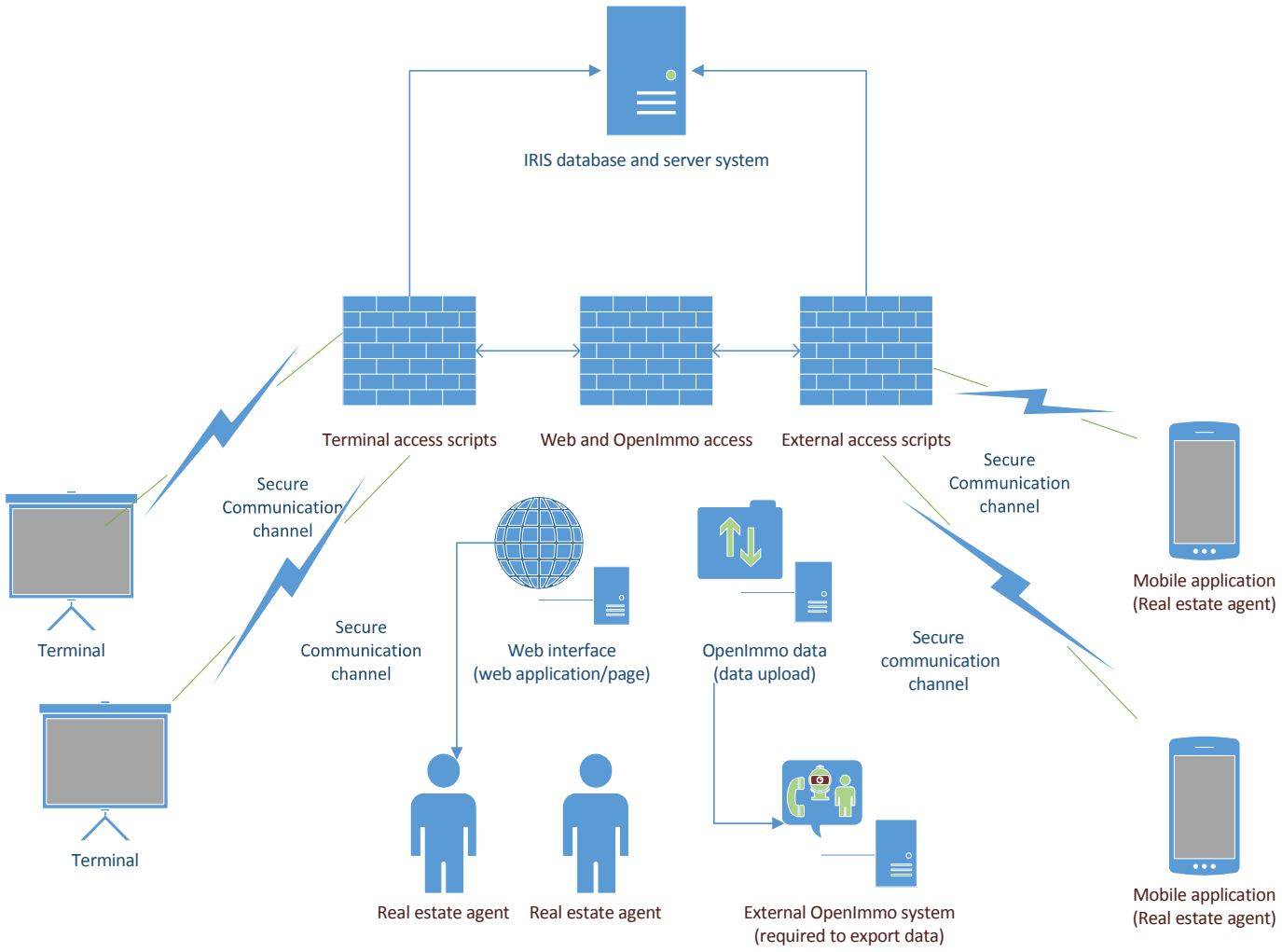
The content of a “Path” element are elements of type “Point” that determines the coordinates. Each Path should include exactly 361 “Point” elements (0-360 degrees). If their number does not reach 361, the particular trajectory is not valid and the application calculates the coordinates.

<*Point*> contains the coordinates (x, y, z) for an angle (0-360). A Point element has four attributes and no space. The first attribute is called "angle" and it stores the angle which coordinates are described by the other attributes. Only non-negative integer numbers are accepted for "angle". The other three attributes are the coordinates ("x", "y", "z"). They are read by the application as float numbers.

# 2. Introduction of Real Estate Portal IRIS

# 2.1 IRIS ecosystem

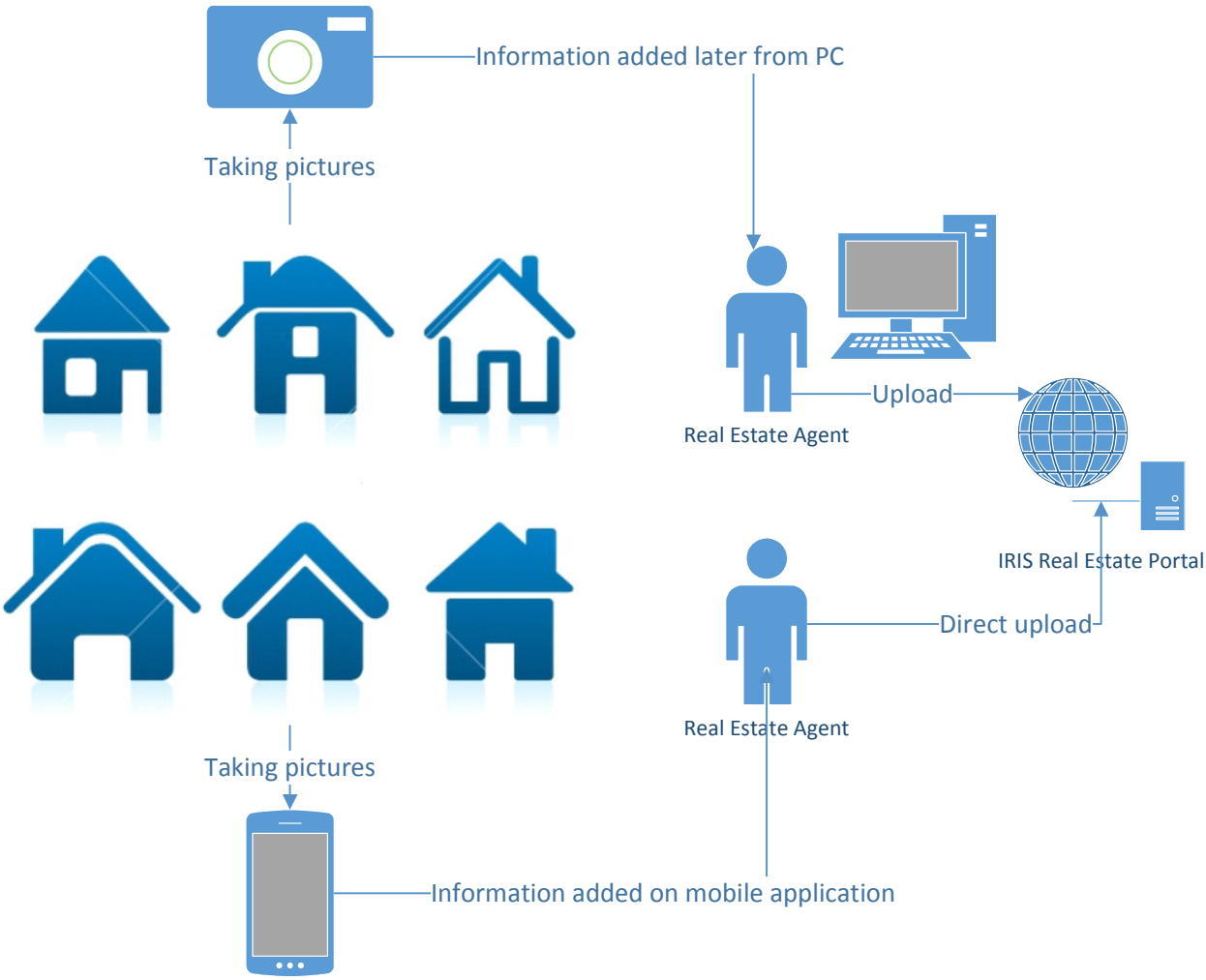
IRIS terminals are part of a larger ecosystem that includes database servers, web applications, data import interfaces and mobile device applications. Although a terminal can work as a standalone system, making use of its full potential requires to have a network connection and the ability to exchange information with other components as shown on the figure below.




Therefore a terminal should be configured to establish and use a secure communication channel to terminal access scripts. These scripts provide a secure access to IRIS database system. Web interface and OpenImmo import scripts interact also with the main database to ensure data integrity.

## 2.1.1 Creating digital content through web interface

The IRIS system allows to add and modify digital content (e.g. offers) and their properties through convenient web interface. In this case the process of adding offers is shown on the diagram below:




There are two possibilities to add information about a real estate offer:

- manually inserting the offer through web interface

In this scenario user needs to login on IRIS web page (www.wave-five.com) and fill-in offer description elements. Picture and video files that have been taken in advance can be uploaded and assigned to newly created offer.

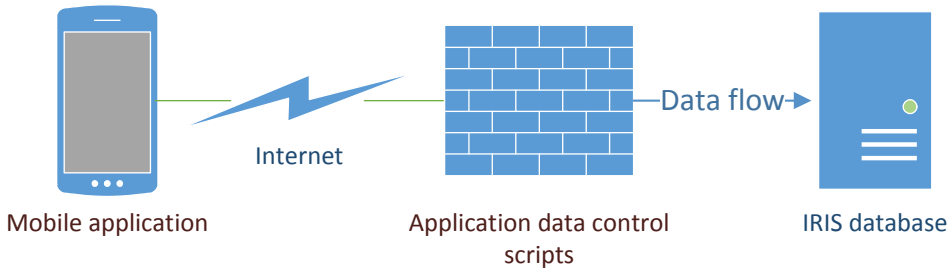
- creating and modifying a real estate offer that has been created with IRIS mobile application

In this scenario, real estate agent phone is used to temporary hold important object information and media files associated with a given real estate. Once

enough information has been collected there it can be exported automatically to IRIS database and offer will be created automatically.

## 2.1.2 Creating digital contents through IRIS mobile application

IRIS mobile application supports Android smartphones and makes it possible to dramatically reduce the effort of collecting information, pictures and video files used to describe a real estate. Collected information can be later on exported to IRIS database and displayed on terminal screen.




Mobile application has the following features:

- Describing real estate objects
- Making pictures of real estate objects
- Making video files of real estate objects

Informaion is kept locally on the smartphone until it is successfully exported to IRIS database server. Depending on customer preferences, real estate objects can be kept (archived) in smartphone internal storage even after they have been exported to the database server.

Each real estate object can have arbitrary number of pictures and video files assigned to it. However since web page limits the number of assigned video and picture files to ten that means that after export only the first ten items will be used.

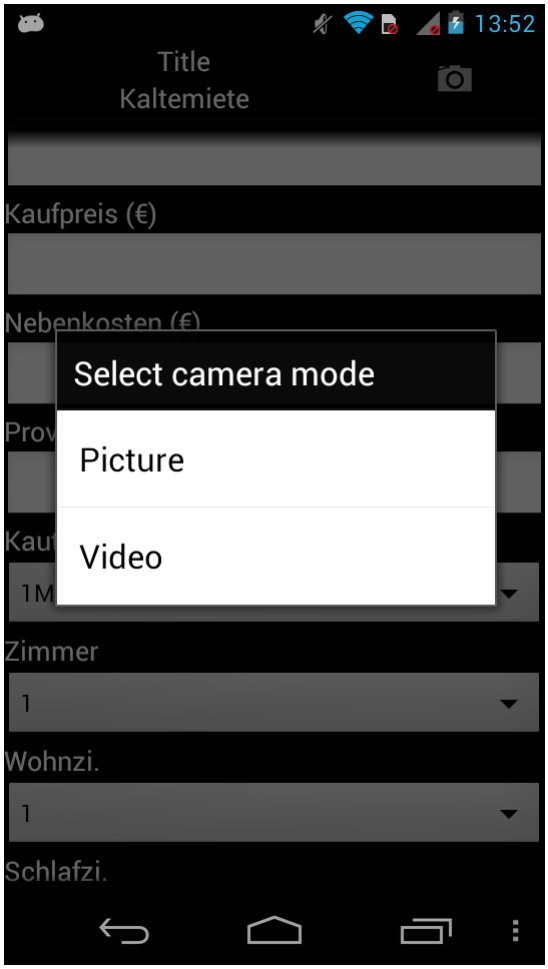
Mobile application can be installed from an APK file available at:

http://www.wave-five.com:3000/scripts/Real%20Estate.apk

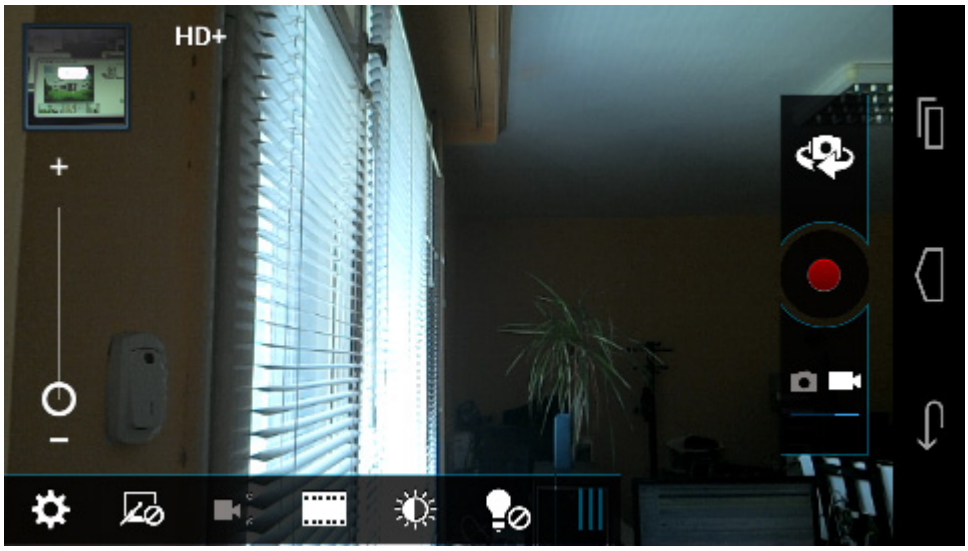
Users should be aware that although applicaiton can be downloaded and installed without any required login information, in order to export data to IRIS

server they would need to configure it and provide valid user name and password.

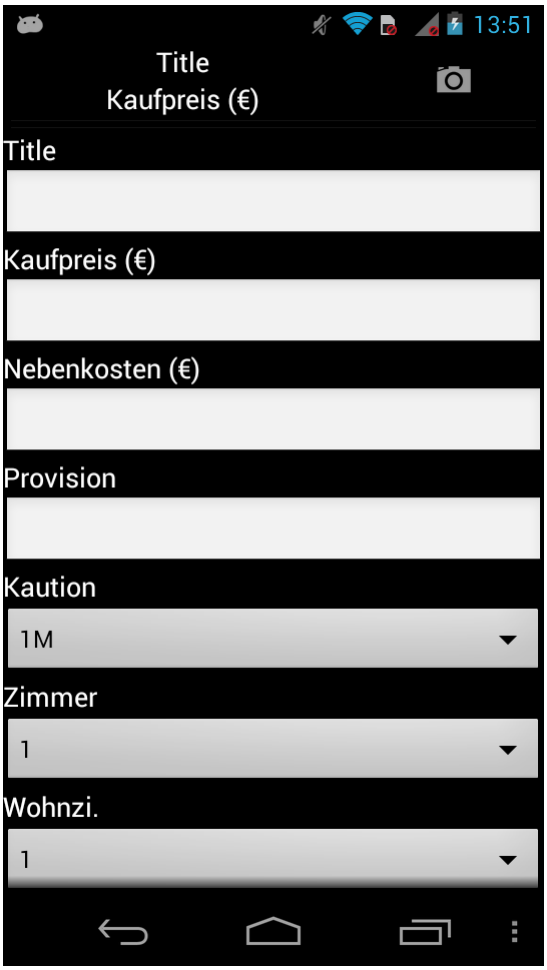
Mobile application makes use of smartphone camera to create pictures and video files.




Therefore picture and video quality depends on smartphone features but usually it is high enough to represent correctly real estate natural features.

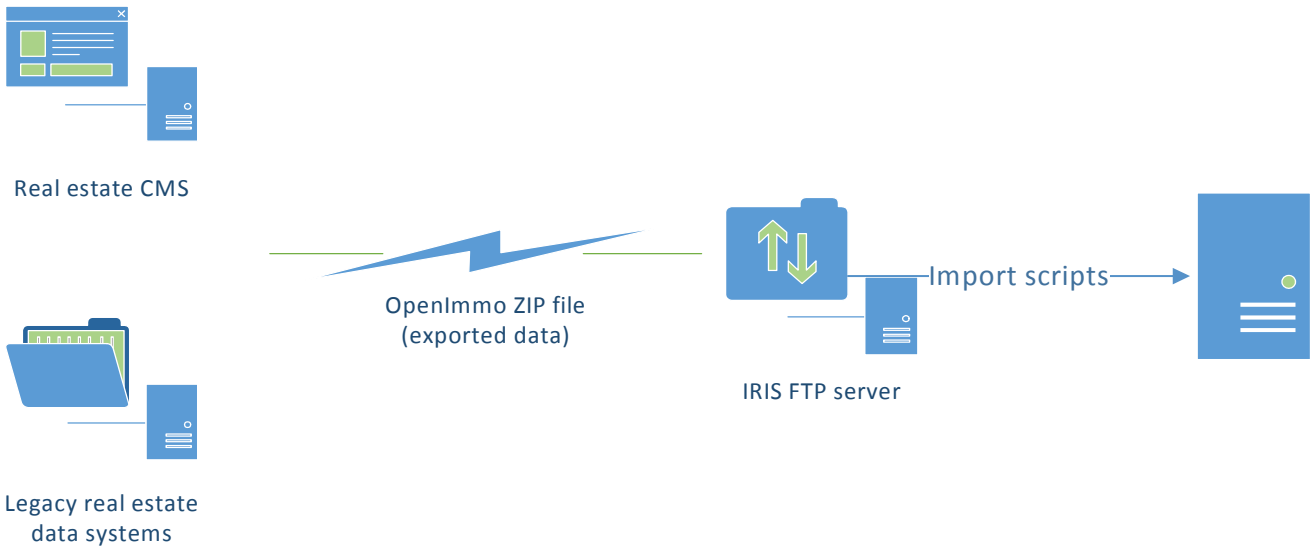



Real estate offers that have not been uploaded, can be modified also from mobile application (however once they are uploaded, they can be modified only from IRIS web site). A typical offer editing screen is shown below.

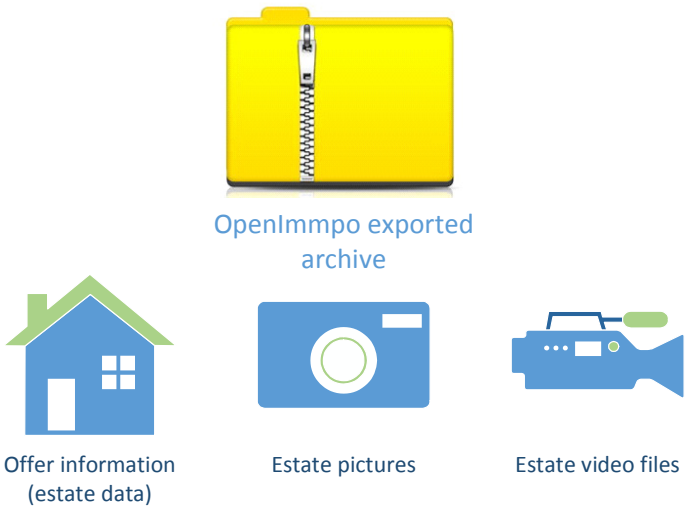


## 2.1.3 Adding information through OpenImmo import

IRIS system also allows to import existing data from OpenImmo standard archive files. Data transfer can be accomplished either through FTP or HTTP services with version 1.0 of application scripts supporting only FTP.




OpenImmo import functionality allows to connect IRIS terminals to different legacy systems and fill in offer data automatically.




IRIS import function expects to receive an OpenImmo archive file containing estate information (XML data) and media files (pictures and video files).

During import only data relevant to IRIS database entries is kept. Any additional information that is available but cannot be stored into IRIS database is dis-

carded. In case of a successful upload, original archive file is removed from FTP location and stored in a backup folder for keeping track of information flow.

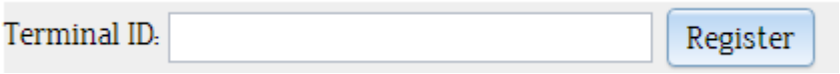
## 2.1.4 How to work with web management interface

Web management interface for terminal system provides a lot of services that make it much easier to set up and modify terminal settings. In order to get access to the web management interface, one should have a valid login data (user name and password) for the IRIS web portal. This login data can be used also to access the full functionality of the web management interface:

- Register and Config

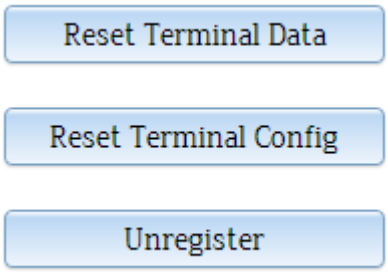
In order to remotely manage a terminal system it has to be registered for a specific user. Since one user can manage multiple terminals, settings can be modified for each of them separately.

Registration of a terminal is done by entering its unique identification number and pressing the “Register” button.




Once a terminal identifier is registered different properties like name and location can be assigned to it. These properties are used in order to be able to easily recognize and identify the terminal with human-readable strings. Terminal properties can be changed dynamically and are automatically downloaded and applied to the respective terminal device. Therefore it is important to type in correctly the termianal identifier - otherwise the system will not be able to correctly match the physical terminal device.

There are three main configuration functions available for each terminal:



- Reset Terminal Data function is used when it is necessary to initiate a complete download of diginal contents assigned to this terminal. Therefore this function should be used only when its necessary.
- Reset Terminal Config function is used when it is necessary to wipe-out old terminal configuration and trigger a download of configuration currently set through the web management interface. Therefore this function should be used with care and only when its necessary.
- Unregister terminal function will remove registration of the currently selected terminal. In case the terminal needs to be accessed again through the web management interface it has to be registered again as described above.

• Assign Properties

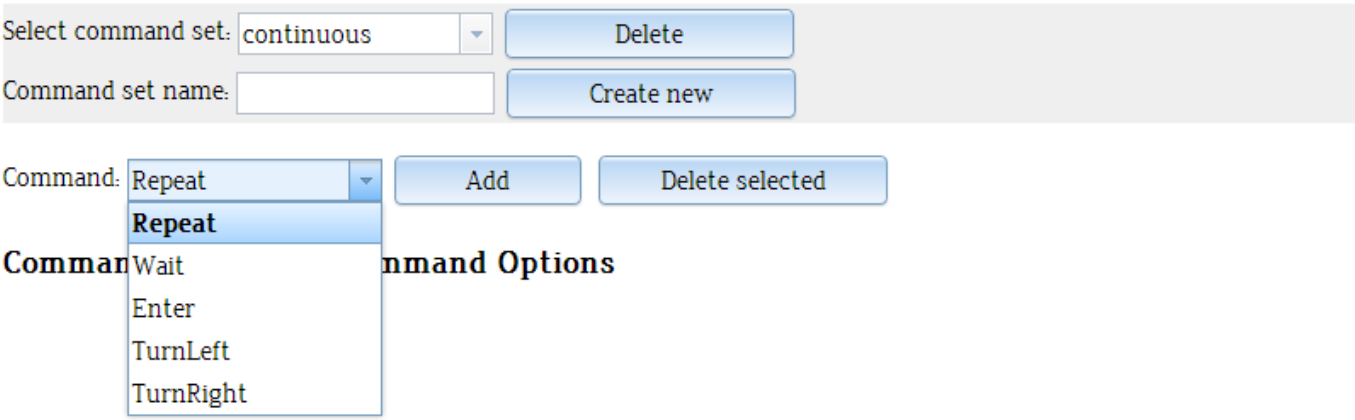
This functional set allows to configure digital contents filter to be assigned to currently selected terminal.


The period selection combo box allows to show only those of the available entries that fit a pre-defined time period. This is only done for convenience and to ease selection. If one wants to have a more complete list of objects assigned to his IRIS portal then timing period can be extended to 1 year.

Only items that have been explicitly selected will be downloaded and shown on the terminal. To avoid misunderstanding of selection terms please note that "Show estates from" combo box refers only to the management interface. Only properties that have been checked are downloaded to the terminal. In case there is nothing cheched on the list, digital contents to download are selected in random way.

- Commands

"Commands" functions let user specify a command set to be used by a terminal. Command sequence can be either pre-defined or user-defined sequence.




Each terminal is delivered with three pre-defined command sets:

- demo - iterates through all available digital content entries and shows them one after another. For each shown entry, all available pictures and video files are shown in a row. Demo command set keeps going through available digital content entries untill application is stopped (or command set is changed).
- continuous - iterates through all available digital content entries and shows them without putting focus on each of them. This means that when an entry comes to a top-front position it is not zoomed to full screen and its video and picture files are not shown in a row.
- single - it brings the first available digital content entry and starts showing in an endless loop its video and picture files.

Depending on user requirements, a custom command set can be created and loaded to IRIS terminals. Custom command sets can be built by combining five different commands:

- *Repeat* defines a loop that can hold other commands
- *Wait* defines a wait block of specific duration
- *Enter* defines a command to bring current top-front entry to full screen

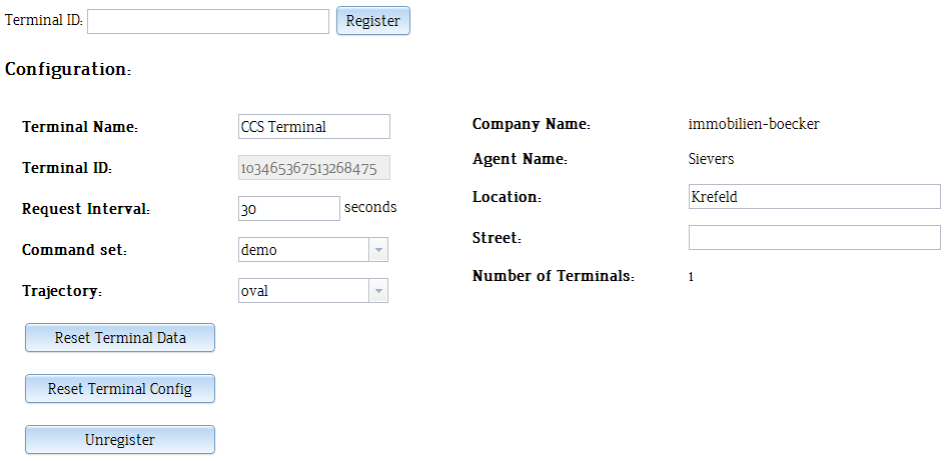
- *TurnLeft* defines a request to rotate carousel one position to the left (clock-wise)
- *TurnRight* defines a command to rotate carousel one position to the right (counter clock-wise)

## 2.1.5 Setting up and modifying terminal personalization

Terminal personalization are typically set prior to delivering the terminal to a customer. However later on, they can be modified and automatically reloaded by terminal software throught a dedicated web application that is available at:

http://www.wave-five.com/cnfms

This application provides a list of all terminals that are registered for a user (terminals can have different physical locations) and allows to modify their configuration parameters.

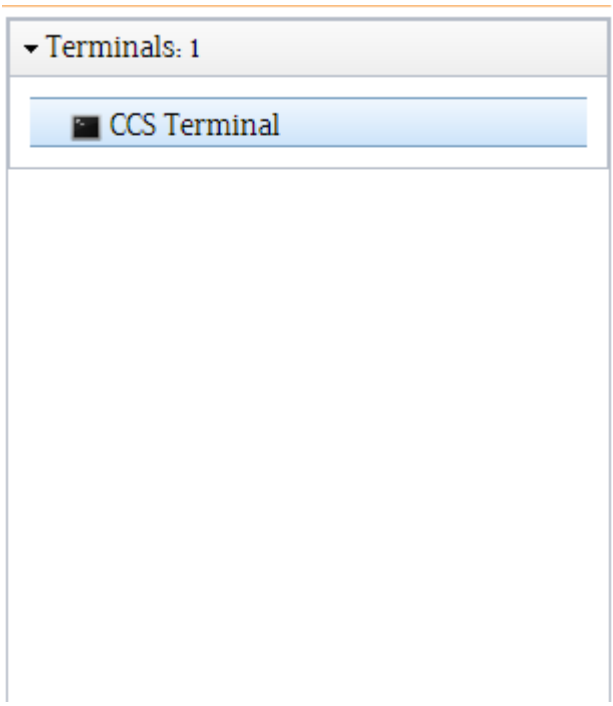



Typical configuration parameters are:

- **Request interval** - the interval of time at which to check for data updates
- **Command set** - example sequence of commands executed while showing automatically different offers (e.g. different real estate offers).
- **Trajectory** - selection of available trajectories that should be used when moving different objects on terminal screen

Terminal configuration interface also allows to modify list of terminals assigned to a particular account (e.g. add, modify or remove existing terminals). This is particularly useful when a customer has a large number of terminals and would

like to assign them a particular roles (for example serving information provided by one employee only) or to assign them to a particular offer.

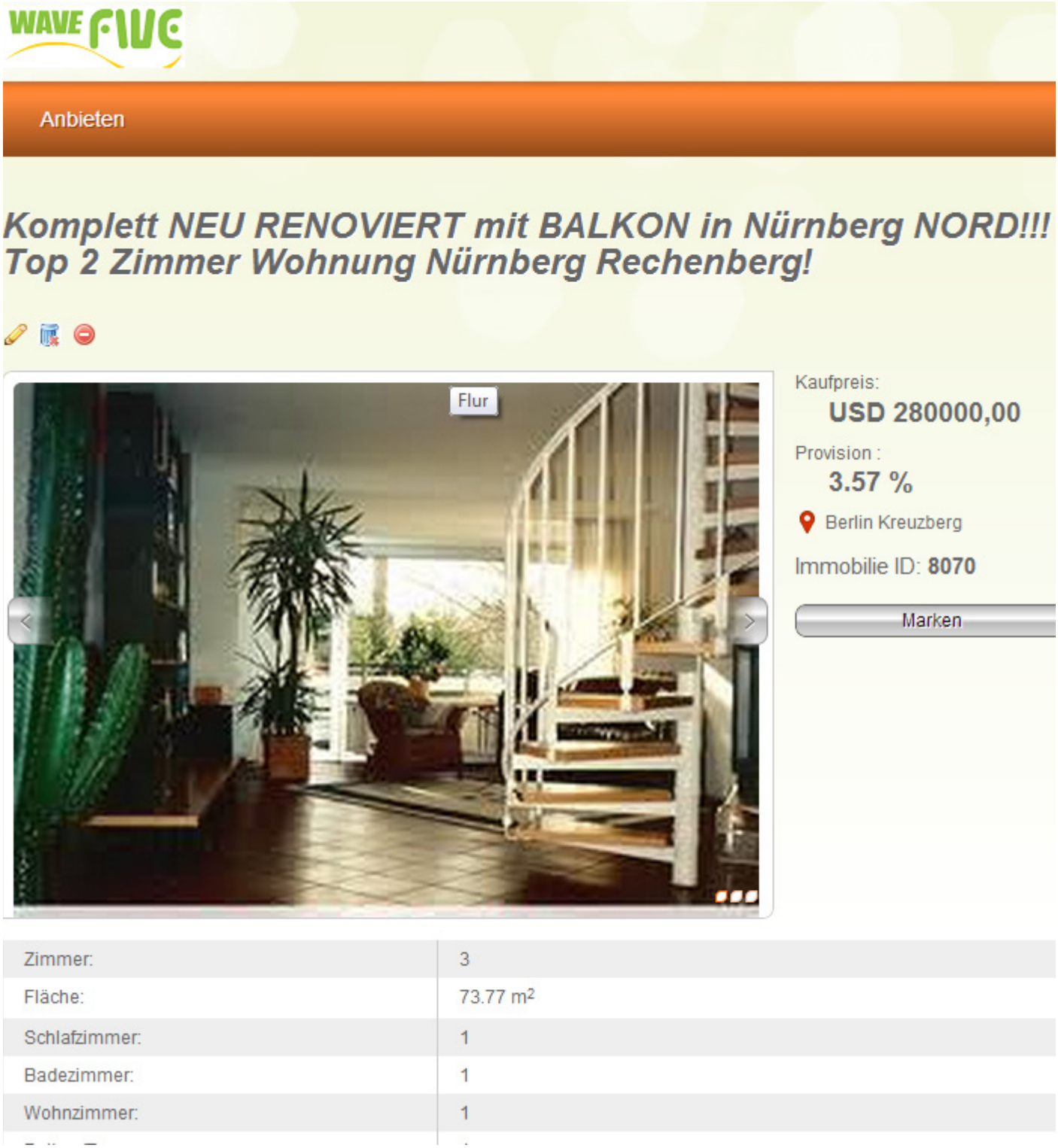



Terminals are identified by their unique ID (as described at the beginning of this document) but they can also be given a human readable name and an address, which makes it possible to keep track on installed terminals and associate them with different digital contents.

In order to reconfigure a terminal please refer to this the opening of this section or to respective “How to ...” items in Chapter 4.


# 3. Working with existing digital contents

Real estate offers frequenty need to be modified or even deleted. This can be done from the IRIS web application, as shown below:




Editing digital content is accessible via dedicated icon.

Editing an offer may be necessary not only to fix a mistyped or wrong figure in offer description but also due to the following important reasons:

- Marking offer as non-visible on the terminal

This can be done by using the “Don’t show” icon on the icon tab:

Changing an offer visibility will trigger also an update of terminal data, which means that file changes will be triggered next time terminal application checks for offer updates.

- Changing picture ordering for an offer

Changing picture ordering is important to make it possible to show new pictures or more informative pictures first.


Picture ordering can be changed in one direction only - by bringing a specifc picture “up” (e.g. to a position closer to the top/front).

- Deleting an offer

The function “Deleting an offer” allows to keep only those offers that are still active. There is an important difference between deleting and offer (which

means that it is completely removed from the database) and changing its visibility (which means that the offer remains in the database but is not shown on the terminal screen).

**Note: Once an offer has been deleted, it can´t be reconstituted.**

# 4. Miscellaneous

# 4.1 How to force an update of terminal application

Terminal applications are designed to automatically check for updates and install them. However in some rare cases a system administrator may need to update the application manually.

There are two possibilies to trigger the update process:

- Forcing CTRL application to do the updates

This method relies on the fact that when started CTRL application automatically does a check for available updates. Therefore the easiest way to trigger an update check is to force stop and restart WaveFiveController application. In order to force stop an application please use “Force stop” Android function available from the “*System settings*” menu of the Android and then “*Apps*” configuration menu in Android.

- Updating manually

Applications can also be updated manually by force stopping both WaveFiveController and CarouselDemo applications from “*Settings menu*” in Android and then “*Apps*” configuration menu. Then applications can be uninstalled manually from this menu and re-downloaded from the following URLs:

- www.wave-five.com/scripts/wavefivecontroller.apk - for CTRL application
- www.wave-five.com/scripts/CarouselDemo.apk - for Carousel application

Packets can be downloaded with default Android browser and installed by double clicking on them. Android package manager recognizes the application type and initiates installation process automatically.

# 4.2 How to force an update of application data

Application data is automatically updated when this is necessary - in case of new offers, offer modification (including offer deletion) and/or modification in configuration data. However in some rare cases a system administrator may need to trigger a data update manually.

In order to trigger full data update please follow these steps:

1. Stop CTRL application (from Settings -> Apps menu)
2. Stop CarouselDemo application (from Settings -> Apps menu)
3. Start a file explorer (Terminals come with preinstalled explorer package)
4. Delete /mnt/sdcard/Demo
5. Log in to the configuration panel (www.wave-five.com/cnfms) and select “Reset Terminal data” button in “Reset and Config” section.
6. Start CarouselDemo application (from Settings -> Apps menu)
7. Start CTRL application (from Settings -> Apps menu)

At this point Carousel application should contact the server again and download all data. Kindly note that in order to complete the download process, internet connection should be available. Depending on the amount of data downloading can take long time and download progress is communicated through an animated GIF shown on the terminal screen:

# 4.3 Known limitations

The following limitations are known and enforced by planning and design decisions:

Table 1: Known limitations

| Module | Description |
|---|---|
| IRIS web application | Each digital content entry can have up to 10 pictures and up to 10 video files. |
| IRIS web application | Each user account should have a valid email address assigned to it. One email can be assigned to multiple accounts. |
| Carousel graphical application | Each entry can display up to 14 textual and numerical properties. Each visible digital content property (text or number) consists of a single line, new line characters are not used and not handled by the Carousel application. |
| Carousel graphical application config | Each path description should contain exactly 361 points. Path descriptions with less than 361 points are considered invalid. |
| Controller application config | Controller application configuration cannot be updated remotely and requires to update the application itself. Update is done automatically when new version of controller application is released |

Known limitations can be released on demand, as they relate to current design and optimizations implemented to reduce code complexity and concentrate mainly on core features.

# 5. Appendix A - Carousel GUI

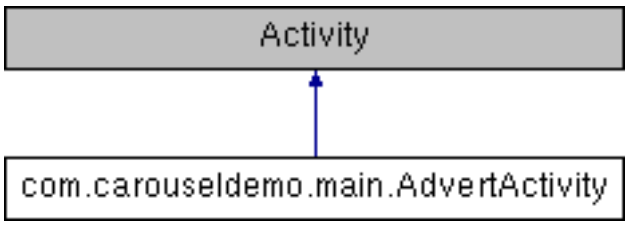
# Technical reference information

This section provides in-depth information on some core IRIS components and in particular Carousel GUI application. End users are not expected to take care about programming details and can safely skip this part.

# Class Documentation

## com.carouseldemo.main.AdvertActivity Class Reference

Inheritance diagram for com.carouseldemo.main.AdvertActivity:




### Public Member Functions

- boolean **onKeyUp** (int keyCode, KeyEvent event)
- void **onBackPressed** ()
- void **onCreate** (Bundle savedInstanceState)
- boolean **playVideo** ()

### Static Public Member Functions

- static void **performCommand** (int command)

### Static Public Attributes

- static **AdvertActivity mActivity** = null
- static String **videoPath** = null

### Private Attributes

- VideoView **mVideoView** = null
- boolean **videoPlaying** = false
- boolean **isBackPressed** = false
- LinearLayout **itemLayout** = null
- long **freeSize** = 0L
- long **totalSize** = 0L
- long **usedSize** = -1L

### Static Private Attributes

- static Drawable **background** = null
- static final String **TAG** = AdvertActivity.class.getSimpleName()

---

### Member Function Documentation

**void com.carouseldemo.main.AdvertActivity.onBackPressed ()**
**void com.carouseldemo.main.AdvertActivity.onCreate (Bundle *savedInstanceState*)**
**boolean com.carouseldemo.main.AdvertActivity.onKeyUp (int *keyCode*, KeyEvent *event*)**
**static void com.carouseldemo.main.AdvertActivity.performCommand (int *command*)** `[static]`
**boolean com.carouseldemo.main.AdvertActivity.playVideo ()**

---

## Member Data Documentation

**Drawable com.carouseldemo.main.AdvertActivity.background = null** `[static], [private]`
**long com.carouseldemo.main.AdvertActivity.freeSize = 0L** `[private]`
**boolean com.carouseldemo.main.AdvertActivity.isBackPressed = false** `[private]`
**LinearLayout com.carouseldemo.main.AdvertActivity.itemLayout = null** `[private]`
**AdvertActivity com.carouseldemo.main.AdvertActivity.mActivity = null** `[static]`
**VideoView com.carouseldemo.main.AdvertActivity.mVideoView = null** `[private]`
**final String com.carouseldemo.main.AdvertActivity.TAG = AdvertActivity.class.getSimpleName()** `[static], [private]`

Tag for a class logging

**long com.carouseldemo.main.AdvertActivity.totalSize = 0L** `[private]`
**long com.carouseldemo.main.AdvertActivity.usedSize = -1L** `[private]`
**String com.carouseldemo.main.AdvertActivity.videoPath = null** `[static]`
**boolean com.carouseldemo.main.AdvertActivity.videoPlaying = false** `[private]`

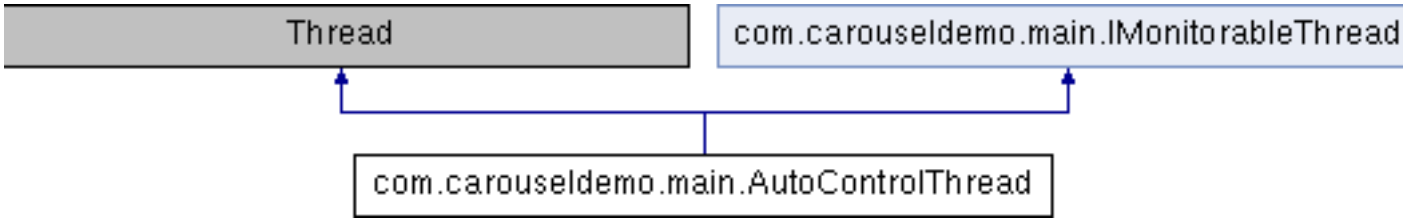
---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/main/**AdvertActivity.java**

# com.carouseldemo.main.AutoControlThread Class Reference

Inheritance diagram for com.carouseldemo.main.AutoControlThread:




## Public Member Functions

- synchronized void **pleaseStop** ()
- synchronized boolean **isStopped** ()
- void **setThreadMonitor** (**IThreadMonitor** mon)
- void **run** ()
- void **pleasePause** ()
- void **pleaseResume** ()
- boolean **isPaused** ()

## Public Attributes

- **Command c** = null
- boolean **shouldWork** = true
- Handler **handler** = new Handler()
- **IThreadMonitor mMon** = null

## Private Attributes

- Integer **flag** = null
- Object **mPauseLock** = new Object()
- boolean **mPaused** = false

## Static Private Attributes

- static final String **TAG** = AutoControlThread.class.getSimpleName()

---

## Member Function Documentation

**boolean com.carouseldemo.main.AutoControlThread.isPaused ()**

Implements **com.carouseldemo.main.IMonitorableThread** (*p.*).

**synchronized boolean com.carouseldemo.main.AutoControlThread.isStopped ()**

**void com.carouseldemo.main.AutoControlThread.pleasePause ()**

Call this on pause.

**void com.carouseldemo.main.AutoControlThread.pleaseResume ()**

Call this on resume.

**synchronized void com.carouseldemo.main.AutoControlThread.pleaseStop ()**

**void com.carouseldemo.main.AutoControlThread.run ()**

**void com.carouseldemo.main.AutoControlThread.setThreadMonitor (IThreadMonitor *mon*)**

Implements **com.carouseldemo.main.IMonitorableThread** (*p.*).

---

## Member Data Documentation

**Command com.carouseldemo.main.AutoControlThread.c = null**
**Integer com.carouseldemo.main.AutoControlThread.flag = null** `[private]`
**Handler com.carouseldemo.main.AutoControlThread.handler = new Handler()**
**IThreadMonitor com.carouseldemo.main.AutoControlThread.mMon = null**
**boolean com.carouseldemo.main.AutoControlThread.mPaused = false** `[private]`
**Object com.carouseldemo.main.AutoControlThread.mPauseLock = new Object()** `[private]`
**boolean com.carouseldemo.main.AutoControlThread.shouldWork = true**
**final String com.carouseldemo.main.AutoControlThread.TAG =**
**AutoControlThread.class.getSimpleName()** `[static], [private]`

Tag for a class logging

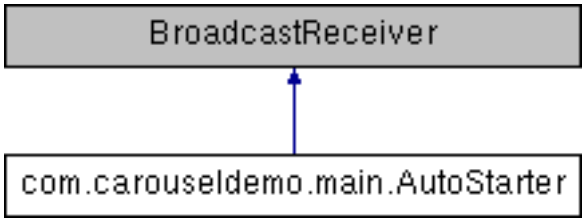
---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/main/**AutoControlThread.java**

## com.carouseldemo.main.AutoStarter Class Reference

Inheritance diagram for com.carouseldemo.main.AutoStarter:




### Public Member Functions

- void **onReceive** (Context arg0, Intent arg1)

---

### Member Function Documentation

**void com.carouseldemo.main.AutoStarter.onReceive (Context *arg0*, Intent *arg1*)**

---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/main/**AutoStarter.java**

# com.carouseldemo.main.BuildConfig Class Reference

## Static Public Attributes

- static final boolean **DEBUG** = true

---

## Member Data Documentation

**final boolean com.carouseldemo.main.BuildConfig.DEBUG = true** `[static]`

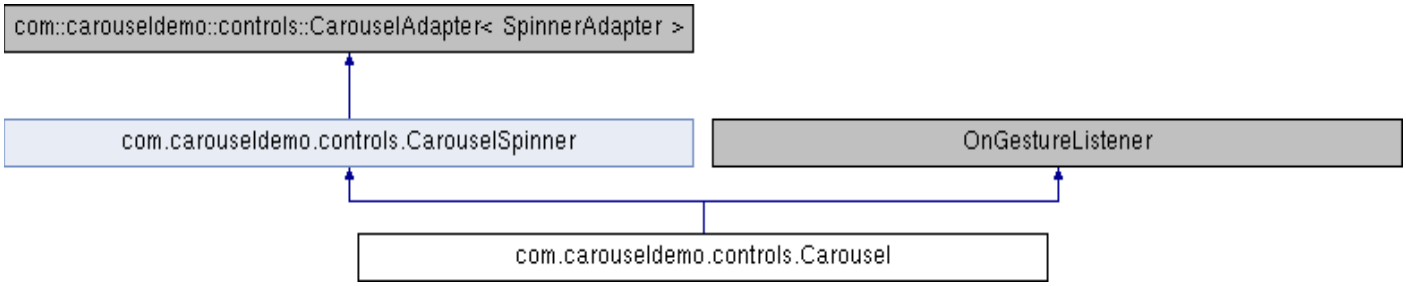
---

**The documentation for this class was generated from the following file:**

- gen/com/carouseldemo/main/**BuildConfig.java**

# com.carouseldemo.controls.Carousel Class Reference

Inheritance diagram for com.carouseldemo.controls.Carousel:




## Classes

- class **FlingRotateRunnable**
- class **ImageAdapter**

## Public Member Functions

- **Carousel** (Context context)
- **Carousel** (Context context, AttributeSet attrs)
- String **getImagePathFromId** (long id)
- **Carousel** (Context context, AttributeSet attrs, int defStyle)
- boolean **showContextMenu** ()
- ViewGroup.LayoutParams **generateLayoutParams** (AttributeSet attrs)
- void **dispatchSetSelected** (boolean selected)
- boolean **showContextMenuForChild** (View originalView)
- boolean **dispatchKeyEvent** (KeyEvent event)
- boolean **onDown** (MotionEvent e)
- boolean **onFling** (MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)
- void **onLongPress** (MotionEvent e)
- boolean **onScroll** (MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)
- boolean **onSingleTapUp** (MotionEvent e)
- void **onShowPress** (MotionEvent e)
- void **scrollToChild** (int i)
- void **setCallbackDuringFling** (boolean shouldCallback)
- void **setCallbackOnUnselectedItemClick** (boolean shouldCallback)
- void **setGravity** (int gravity)

## Public Attributes

- **ImageAdapter mImageAdapter**

## Protected Member Functions

- int **computeHorizontalScrollExtent** ()
- int **computeHorizontalScrollOffset** ()
- int **computeHorizontalScrollRange** ()
- ContextMenuInfo **getContextMenuInfo** ()
- void **onFocusChanged** (boolean gainFocus, int direction, Rect previouslyFocusedRect)
- boolean **checkLayoutParams** (ViewGroup.LayoutParams p)
- ViewGroup.LayoutParams **generateLayoutParams** (ViewGroup.LayoutParams p)
- void **dispatchSetPressed** (boolean pressed)
- int **getChildDrawingOrder** (int childCount, int i)

·boolean **getChildStaticTransformation** (View child, Transformation transformation)
·void **onLayout** (boolean changed, int l, int t, int r, int b)

## Private Member Functions

·void **Calculate3DPosition** (**CarouselItem** child, int diameter, float angleOffset)
·int **calculateTop** (View child, boolean duringLayout)
·boolean **dispatchLongPress** (View view, int position, long id)
·void **dispatchUnpress** ()
·int **getCenterOfGallery** ()
·void **makeAndAddView** (int position, float angleOffset)
·void **onFinishedMovement** ()
·void **scrollIntoSlots** ()
·void **setUpChild** (**CarouselItem** child, int index, float angleOffset)
·void **updateSelectedItemMetadata** ()

## Static Private Member Functions

·static int **getCenterOfView** (View view)

## Private Attributes

·AdapterContextMenuInfo **mContextMenuInfo**
·int **mAnimationDuration** = 900
·Camera **mCamera** = new Camera()
·Hashtable< Float, **CarouselPoint** > **trajectory** = null
·Runnable **mDisableSuppressSelectionChangedRunnable**
·int **mDownTouchPosition**
·View **mDownTouchView**
·**FlingRotateRunnable mFlingRunnable** = new **FlingRotateRunnable**()
·GestureDetector **mGestureDetector**
·int **mGravity**
·boolean **mIsFirstScroll**
·View **mSelectedChild**
·boolean **mShouldCallbackDuringFling** = true
·boolean **mShouldCallbackOnUnselectedItemClick** = true
·boolean **mSuppressSelectionChanged**
·float **mTheta** = (float) (15.0f * (Math.PI / 180.0))
·long **timeMs** = 0
·boolean **add** = false
·int **transitionFront** = 0
·int **previousMinZ** = -1
·int **lastMinZ** = -1

## Static Private Attributes

·static final String **TAG** = Carousel.class.getSimpleName()
·static final boolean **localLOGV** = false
·static final float **MAX_THETA** = 45.0f
·static final int **SCROLL_TO_FLING_UNCERTAINTY_TIMEOUT** = 250
·static final int **HORIZONTAL_TRAJECTORY** = 1
·static final int **VERTICAL_TRAJECTORY** = 2
·static final int **OVAL_TRAJECTORY** = 3
·static int **trajectoryOrientation** = **HORIZONTAL_TRAJECTORY**
·static boolean **calucateCoordinates** = false

## Detailed Description

**Author:**

Isapov & Ivanov

## Constructor & Destructor Documentation

**com.carouseldemo.controls.Carousel.Carousel (Context *context*)**
**com.carouseldemo.controls.Carousel.Carousel (Context *context*, AttributeSet *attrs*)**
**com.carouseldemo.controls.Carousel.Carousel (Context *context*, AttributeSet *attrs*, int *defStyle*)**

## Member Function Documentation

**void com.carouseldemo.controls.Carousel.Calculate3DPosition (CarouselItem *child*, int *diameter*, float *angleOffset*)** `[private]`

**int com.carouseldemo.controls.Carousel.calculateTop (View *child*, boolean *duringLayout*)** `[private]`

Figure out vertical placement based on mGravity

**Parameters:**

| | |
|---|---|
| *child* | Child to place |

**Returns:**

Where the top of the child should be

**boolean com.carouseldemo.controls.Carousel.checkLayoutParams (ViewGroup.LayoutParams *p*)** `[protected]`

**int com.carouseldemo.controls.Carousel.computeHorizontalScrollExtent ()** `[protected]`

Compute the horizontal extent of the horizontal scrollbar's thumb within the horizontal range. This value is used to compute the length of the thumb within the scrollbar's track.

**int com.carouseldemo.controls.Carousel.computeHorizontalScrollOffset ()** `[protected]`

Compute the horizontal offset of the horizontal scrollbar's thumb within the horizontal range. This value is used to compute the position of the thumb within the scrollbar's track.

**int com.carouseldemo.controls.Carousel.computeHorizontalScrollRange ()** `[protected]`

Compute the horizontal range that the horizontal scrollbar represents.

**boolean com.carouseldemo.controls.Carousel.dispatchKeyEvent (KeyEvent *event*)**

**boolean com.carouseldemo.controls.Carousel.dispatchLongPress (View *view*, int *position*, long *id*)** `[private]`

**void com.carouseldemo.controls.Carousel.dispatchSetPressed (boolean *pressed*)** `[protected]`

**void com.carouseldemo.controls.Carousel.dispatchSetSelected (boolean *selected*)**

**void com.carouseldemo.controls.Carousel.dispatchUnpress ()** `[private]`

**ViewGroup.LayoutParams com.carouseldemo.controls.Carousel.generateLayoutParams (ViewGroup.LayoutParams *p*)** `[protected]`

**ViewGroup.LayoutParams com.carouseldemo.controls.Carousel.generateLayoutParams (AttributeSet *attrs*)**

**int com.carouseldemo.controls.Carousel.getCenterOfGallery ()** `[private]`

**Returns:**

The center of this Gallery.

**static int com.carouseldemo.controls.Carousel.getCenterOfView (View *view*)** `[static], [private]`

**Returns:**

The center of the given view.

**int com.carouseldemo.controls.Carousel.getChildDrawingOrder (int *childCount*, int *i*)** `[protected]`

Index of the child to draw for this iteration

**boolean com.carouseldemo.controls.Carousel.getChildStaticTransformation (View *child*, Transformation *transformation*)** `[protected]`

Transform an item depending on its coordinates

**ContextMenuInfo com.carouseldemo.controls.Carousel.getContextMenuInfo ()** `[protected]`

Implemented to handle touch screen motion events. Extra information about the item for which the context menu should be shown.

**String com.carouseldemo.controls.Carousel.getImagePathFromId (long *id*)**

**void com.carouseldemo.controls.Carousel.makeAndAddView (int *position*, float *angleOffset*)** `[private]`

**boolean com.carouseldemo.controls.Carousel.onDown (MotionEvent *e*)**

**void com.carouseldemo.controls.Carousel.onFinishedMovement ()** `[private]`

Called when rotation is finished

**boolean com.carouseldemo.controls.Carousel.onFling (MotionEvent *e1*, MotionEvent *e2*, float *velocityX*, float *velocityY*)**

**void com.carouseldemo.controls.Carousel.onFocusChanged (boolean *gainFocus*, int *direction*, Rect *previouslyFocusedRect*)** `[protected]`

Handles left, right, and clicking

**See Also:**

android.view.View::onKeyDown

**void com.carouseldemo.controls.Carousel.onLayout (boolean *changed*, int *l*, int *t*, int *r*, int *b*)** `[protected]`

Setting up images after layout changed

**void com.carouseldemo.controls.Carousel.onLongPress (MotionEvent *e*)**

**boolean com.carouseldemo.controls.Carousel.onScroll (MotionEvent *e1*, MotionEvent *e2*, float *distanceX*, float *distanceY*)**

**void com.carouseldemo.controls.Carousel.onShowPress (MotionEvent *e*)**

**boolean com.carouseldemo.controls.Carousel.onSingleTapUp (MotionEvent *e*)**

**void com.carouseldemo.controls.Carousel.scrollIntoSlots ()** `[private]`

Brings an item with nearest to 0 degrees angle to this angle and sets it selected

**void com.carouseldemo.controls.Carousel.scrollToChild (int *i*)**

**void com.carouseldemo.controls.Carousel.setCallbackDuringFling (boolean *shouldCallback*)**

Whether or not to callback on any **getOnItemSelectedListener()** while the items are being flinged. If false, only the final selected item will cause the callback. If true, all items between the first and the final will cause callbacks.

**Parameters:**

| | |
|---|---|
| *shouldCallback* | Whether or not to callback on the listener while the items are being flinged. |

**void com.carouseldemo.controls.Carousel.setCallbackOnUnselectedItemClick (boolean *shouldCallback*)**

Whether or not to callback when an item that is not selected is clicked. If false, the item will become selected (and re-centered). If true, the **getOnItemClickListener()** will get the callback.

**Parameters:**

| | |
|---|---|
| *shouldCallback* | Whether or not to callback on the listener when a item that is not selected is clicked. |

**void com.carouseldemo.controls.Carousel.setGravity (int *gravity*)**

Sets how long the transition animation should run when a child view changes position. Only relevant if animation is turned on.

**Parameters:**

| | |
|---|---|
| *animationDuration Millis* | The duration of the transition, in milliseconds. |

ref android.R.styleable::Gallery_animationDuration

**void com.carouseldemo.controls.Carousel.setUpChild (CarouselItem *child*, int *index*, float *angleOffset*) `[private]`**

Helper for makeAndAddView to set the position of a view and fill out its layout parameters.

**Parameters:**

| | |
|---|---|
| *child* | The view to position |
| *offset* | Offset from the selected position |
| *x* | X-coordintate indicating where this view should be placed. This will either be the left or right edge of the view, depending on the fromLeft paramter |
| *fromLeft* | Are we posiitoning views based on the left edge? (i.e., building from left to right)? |

**boolean com.carouseldemo.controls.Carousel.showContextMenu ()**

Bring up the context menu for this view.

**boolean com.carouseldemo.controls.Carousel.showContextMenuForChild (View *originalView*)**

**void com.carouseldemo.controls.Carousel.updateSelectedItemMetadata () `[private]`**

---

## Member Data Documentation

**boolean com.carouseldemo.controls.Carousel.add = false `[private]`**

**boolean com.carouseldemo.controls.Carousel.calucateCoordinates = false `[static]`, `[private]`**

**final int com.carouseldemo.controls.Carousel.HORIZONTAL_TRAJECTORY = 1 `[static]`, `[private]`**

**int com.carouseldemo.controls.Carousel.lastMinZ = -1 `[private]`**

**final boolean com.carouseldemo.controls.Carousel.localLOGV = false `[static]`, `[private]`**

If logging should be inside class

**int com.carouseldemo.controls.Carousel.mAnimationDuration = 900 `[private]`**

How long the transition animation should run when a child view changes position, measured in milliseconds.

**final float com.carouseldemo.controls.Carousel.MAX_THETA = 45.0f `[static]`, `[private]`**

Default min quantity of images Default max quantity of images Max theta

**Camera com.carouseldemo.controls.Carousel.mCamera = new Camera()** `[private]`

Camera to make 3D rotation

**AdapterContextMenuInfo com.carouseldemo.controls.Carousel.mContextMenuInfo** `[private]`

The info for adapter context menu

**Runnable com.carouseldemo.controls.Carousel.mDisableSuppressSelectionChangedRunnable** `[private]`

```
Initial value:= new Runnable() {
        public void run() {
            mSuppressSelectionChanged = false;
            selectionChanged();
        }
    }
```

Sets mSuppressSelectionChanged = false. This is used to set it to false in the future. It will also trigger a selection changed.

**int com.carouseldemo.controls.Carousel.mDownTouchPosition** `[private]`

The position of the item that received the user's down touch.

**View com.carouseldemo.controls.Carousel.mDownTouchView** `[private]`

The view of the item that received the user's down touch.

**FlingRotateRunnable com.carouseldemo.controls.Carousel.mFlingRunnable = new FlingRotateRunnable()** `[private]`

Executes the delta rotations from a fling or scroll movement.

**GestureDetector com.carouseldemo.controls.Carousel.mGestureDetector** `[private]`

Helper for detecting touch gestures.

**int com.carouseldemo.controls.Carousel.mGravity** `[private]`

Gravity for the widget

**ImageAdapter com.carouseldemo.controls.Carousel.mImageAdapter**

If items should be reflected

**boolean com.carouseldemo.controls.Carousel.mIsFirstScroll** `[private]`

If true, this onScroll is the first for this user's drag (remember, a drag sends many onScrolls).

**View com.carouseldemo.controls.Carousel.mSelectedChild** `[private]`

Set max quantity of images Set min quantity of images If true, we have received the "invoke" (center or enter buttons) key down. This is checked before we action on the "invoke" key up, and is subsequently cleared. The currently selected item's child.

**boolean com.carouseldemo.controls.Carousel.mShouldCallbackDuringFling = true** `[private]`

Whether to continuously callback on the item selected listener during a fling.

**boolean com.carouseldemo.controls.Carousel.mShouldCallbackOnUnselectedItemClick = true** `[private]`

Whether to callback when an item that is not selected is clicked.

**boolean com.carouseldemo.controls.Carousel.mSuppressSelectionChanged** `[private]`

When fling runnable runs, it resets this to false. Any method along the path until the end of its run() can set this to true to abort any remaining fling. For example, if we've reached either the leftmost or rightmost item, we will set this to true. If true, do not callback to item selected listener.

**float com.carouseldemo.controls.Carousel.mTheta = (float) (15.0f * (Math.PI / 180.0))** `[private]`

The axe angle

**final int com.carouseldemo.controls.Carousel.OVAL_TRAJECTORY = 3** `[static], [private]`

**int com.carouseldemo.controls.Carousel.previousMinZ = -1** `[private]`

**final int com.carouseldemo.controls.Carousel.SCROLL_TO_FLING_UNCERTAINTY_TIMEOUT = 250** `[static], [private]`

Duration in milliseconds from the start of a scroll during which we're unsure whether the user is scrolling or flinging.

**final String com.carouseldemo.controls.Carousel.TAG = Carousel.class.getSimpleName()** `[static], [private]`

Tag for a class logging

**long com.carouseldemo.controls.Carousel.timeMs = 0** `[private]`

**Hashtable<Float, CarouselPoint> com.carouseldemo.controls.Carousel.trajectory = null** `[private]`

**int com.carouseldemo.controls.Carousel.trajectoryOrientation = HORIZONTAL_TRAJECTORY** `[static], [private]`

**int com.carouseldemo.controls.Carousel.transitionFront = 0** `[private]`

**final int com.carouseldemo.controls.Carousel.VERTICAL_TRAJECTORY = 2** `[static], [private]`

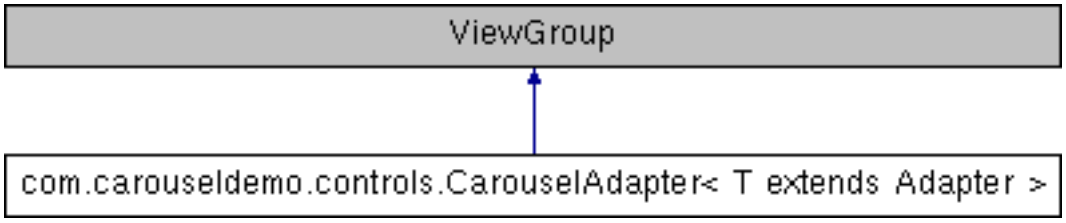
---

**The documentation for this class was generated from the following file:**

· src/com/carouseldemo/controls/**Carousel.java**

# com.carouseldemo.controls.CarouselAdapter< T extends Adapter > Class Reference

Inheritance diagram for com.carouseldemo.controls.CarouselAdapter< T extends Adapter >:




## Classes

- class **AdapterContextMenuInfo**
- class **AdapterDataSetObserver**
- interface **OnItemClickListener**
- interface **OnItemLongClickListener**
- interface **OnItemSelectedListener**
- class **SelectionNotifier**

## Public Member Functions

- **CarouselAdapter** (Context context)
- **CarouselAdapter** (Context context, AttributeSet attrs)
- **CarouselAdapter** (Context context, AttributeSet attrs, int defStyle)
- void **setOnItemClickListener** (OnItemClickListener listener)
- final OnItemClickListener **getOnItemClickListener** ()
- boolean **performItemClick** (View view, int position, long id)
- void **setOnItemLongClickListener** (OnItemLongClickListener listener)
- final OnItemLongClickListener **getOnItemLongClickListener** ()
- void **setOnItemSelectedListener** (OnItemSelectedListener listener)
- final OnItemSelectedListener **getOnItemSelectedListener** ()
- abstract T **getAdapter** ()
- abstract void **setAdapter** (T adapter)
- void **addView** (View child)
- void **addView** (View child, int index)
- void **addView** (View child, LayoutParams params)
- void **addView** (View child, int index, LayoutParams params)
- void **removeView** (View child)
- void **removeViewAt** (int index)
- void **removeAllViews** ()
- .CapturedViewProperty int **getSelectedItemPosition** ()
- .CapturedViewProperty long **getSelectedItemId** ()
- abstract View **getSelectedView** ()
- Object **getSelectedItem** ()
- .CapturedViewProperty int **getCount** ()
- int **getPositionForView** (View view)
- int **getFirstVisiblePosition** ()
- int **getLastVisiblePosition** ()
- abstract void **setSelection** (int position)
- void **setEmptyView** (View emptyView)
- View **getEmptyView** ()

·void **setFocusable** (boolean focusable)
·void **setFocusableInTouchMode** (boolean focusable)
·Object **getItemAtPosition** (int position)
·long **getItemIdAtPosition** (int position)
·void **setOnClickListener** (OnClickListener l)
·boolean **dispatchPopulateAccessibilityEvent** (AccessibilityEvent event)

## Static Public Attributes

·static final int **ITEM_VIEW_TYPE_IGNORE** = -1
·static final int **ITEM_VIEW_TYPE_HEADER_OR_FOOTER** = -2
·static final int **INVALID_POSITION** = -1
·static final long **INVALID_ROW_ID** = Long.MIN_VALUE

## Protected Member Functions

·void **onLayout** (boolean changed, int left, int top, int right, int bottom)
·void **dispatchSaveInstanceState** (SparseArray< Parcelable > container)
·void **dispatchRestoreInstanceState** (SparseArray< Parcelable > container)
·boolean **canAnimate** ()

## Private Member Functions

·void **updateEmptyStatus** (boolean empty)
·void **fireOnSelected** ()

## Private Attributes

·int **mLayoutHeight**
·View **mEmptyView**
·boolean **mDesiredFocusableState**
·boolean **mDesiredFocusableInTouchModeState**
·SelectionNotifier **mSelectionNotifier** = null

---

## Constructor & Destructor Documentation

**com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.CarouselAdapter (Context *context*)**
**com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.CarouselAdapter (Context *context*, AttributeSet *attrs*)**
**com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.CarouselAdapter (Context *context*, AttributeSet *attrs*, int *defStyle*)**

---

## Member Function Documentation

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.addView (View *child*)**

This method is not supported and throws an UnsupportedOperationException when called.

**Parameters:**

| | |
|---|---|
| *child* | Ignored. |

**Exceptions:**

| | |
|---|---|
| *UnsupportedOperationException* | Every time this method is invoked. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.addView (View *child*, int *index*)**

This method is not supported and throws an UnsupportedOperationException when called.

**Parameters:**

| | |
|---|---|
| *child* | Ignored. |
| *index* | Ignored. |

**Exceptions:**

| | |
|---|---|
| *UnsupportedOperationException* | Every time this method is invoked. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.addView (View *child*, LayoutParams *params*)**

This method is not supported and throws an UnsupportedOperationException when called.

**Parameters:**

| | |
|---|---|
| *child* | Ignored. |
| *params* | Ignored. |

**Exceptions:**

| | |
|---|---|
| *UnsupportedOperationException* | Every time this method is invoked. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.addView (View *child*, int *index*, LayoutParams *params*)**

This method is not supported and throws an UnsupportedOperationException when called.

**Parameters:**

| | |
|---|---|
| *child* | Ignored. |
| *index* | Ignored. |
| *params* | Ignored. |

**Exceptions:**

| | |
|---|---|
| *UnsupportedOperationException* | Every time this method is invoked. |

**boolean com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.canAnimate () `[protected]`**

**boolean com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.dispatchPopulateAccessibilityEvent (AccessibilityEvent *event*)**

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.dispatchRestoreInstanceState (SparseArray< Parcelable > *container*) `[protected]`**

Override to prevent thawing of any views created by the adapter.

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.dispatchSaveInstanceState (SparseArray< Parcelable > *container*) `[protected]`**

Override to prevent freezing of any views created by the adapter.

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.fireOnSelected () `[private]`**

**abstract T com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getAdapter () `[pure virtual]`**

Returns the adapter currently associated with this widget.

**Returns:**

The adapter used to provide this view's content.

**.CapturedViewProperty int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getCount ()**

**Returns:**

The number of items owned by the Adapter associated with this CarouselAdapter. (This is the number of data items, which may be larger than the number of visible view.)

**View com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getEmptyView ()**

When the current adapter is empty, the CarouselAdapter can display a special view call the empty view. The empty view is used to provide feedback to the user that no data is available in this CarouselAdapter.

**Returns:**

The view to show if the adapter is empty.

**int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getFirstVisiblePosition ()**

Returns the position within the adapter's data set for the first item displayed on screen.

**Returns:**

The position within the adapter's data set

**Object com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getItemAtPosition (int *position*)**

Gets the data associated with the specified position in the list.

**Parameters:**

| | |
|---|---|
| *position* | Which data to get |

**Returns:**

The data associated with the specified position in the list

**long com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getItemIdAtPosition (int *position*)**

**int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getLastVisiblePosition ()**

Returns the position within the adapter's data set for the last item displayed on screen.

**Returns:**

The position within the adapter's data set

**final OnItemClickListener com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getOnItemClickListener ()**

**Returns:**

The callback to be invoked with an item in this CarouselAdapter has been clicked, or null id no callback has been set.

**final OnItemLongClickListener com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getOnItemLongClickListener ()**

**Returns:**

The callback to be invoked with an item in this CarouselAdapter has been clicked and held, or null id no callback has been set.

**final OnItemSelectedListener com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getOnItemSelectedListener ()**

**int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getPositionForView (View *view*)**

Get the position within the adapter's data set for the view, where view is a an adapter item or a descendant of an adapter item.

**Parameters:**

| | |
|---|---|
| *view* | an adapter item, or a descendant of an adapter item. This must be visible in this CarouselAdapter at the time of the call. |

**Returns:**

the position within the adapter's data set of the view, or **INVALID_POSITION** if the view does not correspond to a list item (or it is not currently visible).

**Object com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getSelectedItem ()**

**Returns:**

The data corresponding to the currently selected item, or null if there is nothing selected.

**.CapturedViewProperty long com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getSelectedItemId ()**

**Returns:**

The id corresponding to the currently selected item, or **INVALID_ROW_ID** if nothing is selected.

**.CapturedViewProperty int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getSelectedItemPosition ()**

Return the position of the currently selected item within the adapter's data set

**Returns:**

int Position (starting at 0), or **INVALID_POSITION** if there is nothing selected.

**abstract View com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.getSelectedView () `[pure virtual]`**

**Returns:**

The view corresponding to the currently selected item, or null if nothing is selected

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.onLayout (boolean *changed*, int *left*, int *top*, int *right*, int *bottom*) `[protected]`**

**boolean com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.performItemClick (View *view*, int *position*, long *id*)**

Call the **OnItemClickListener**, if it is defined.

**Parameters:**

| | |
|---|---|
| *view* | The view within the CarouselAdapter that was clicked. |
| *position* | The position of the view in the adapter. |
| *id* | The row id of the item that was clicked. |

**Returns:**

True if there was an assigned **OnItemClickListener** that was called, false otherwise is returned.

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.removeAllViews ()**

This method is not supported and throws an UnsupportedOperationException when called.

**Exceptions:**

| | |
|---|---|
| *UnsupportedOperationException* | Every time this method is invoked. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.removeView (View *child*)**

This method is not supported and throws an UnsupportedOperationException when called.

**Parameters:**

| | |
|---|---|
| *child* | Ignored. |

**Exceptions:**

| | |
|---|---|
| *UnsupportedOperationException* | Every time this method is invoked. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.removeViewAt (int *index*)**

This method is not supported and throws an UnsupportedOperationException when called.

**Parameters:**

| | |
|---|---|
| *index* | Ignored. |

**Exceptions:**

| | |
|---|---|
| *UnsupportedOperationException* | Every time this method is invoked. |

**abstract void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setAdapter (T *adapter*) `[pure virtual]`**

Sets the adapter that provides the data and the views to represent the data in this widget.

**Parameters:**

| | |
|---|---|
| *adapter* | The adapter to use to create this view's content. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setEmptyView (View *emptyView*)**

Sets the view to show if the adapter is empty

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setFocusable (boolean *focusable*)**

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setFocusableInTouchMode (boolean *focusable*)**

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setOnClickListener (OnClickListener *l*)**

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setOnItemClickListener (OnItemClickListener *listener*)**

Register a callback to be invoked when an item in this CarouselAdapter has been clicked.

**Parameters:**

| | |
|---|---|
| *listener* | The callback that will be invoked. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setOnItemLongClickListener (OnItemLongClickListener *listener*)**

Register a callback to be invoked when an item in this CarouselAdapter has been clicked and held

**Parameters:**

| | |
|---|---|
| *listener* | The callback that will run |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setOnItemSelectedListener (OnItemSelectedListener *listener*)**

Register a callback to be invoked when an item in this CarouselAdapter has been selected.

**Parameters:**

| | |
|---|---|
| *listener* | The callback that will run |

**abstract void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.setSelection (int *position*)** `[pure virtual]`

Sets the currently selected item. To support accessibility subclasses that override this method must invoke the override super method first.

**Parameters:**

| | |
|---|---|
| *position* | Index (starting at 0) of the data item to be selected. |

**void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.updateEmptyStatus (boolean *empty*)** `[private]`

Update the status of the list based on the empty parameter. If empty is true and we have an empty view, display it. In all the other cases, make sure that the listview is VISIBLE and that the empty view is GONE (if it's not null).

## Member Data Documentation

**final int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.INVALID_POSITION = -1** `[static]`

Represents an invalid position. All valid positions are in the range 0 to 1 less than the number of items in the current adapter.

**final long com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.INVALID_ROW_ID = Long.MIN_VALUE** `[static]`

Represents an empty or invalid row id

**final int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.ITEM_VIEW_TYPE_HEADER_OR_FOOTER = -2** `[static]`

The item view type returned by **Adapter#getItemViewType(int)** when the item is a header or footer.

**final int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.ITEM_VIEW_TYPE_IGNORE = -1** `[static]`

The item view type returned by **Adapter#getItemViewType(int)** when the adapter does not want the item's view recycled.

**boolean com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.mDesiredFocusableInTouchModeState** `[private]`

**boolean com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.mDesiredFocusableState** `[private]`

Indicates what focusable state is requested when calling **setFocusable()**. In addition to this, this view has other criteria for actually determining the focusable state (such as whether its empty or the text filter is shown).

**See Also:**

**setFocusable(boolean)**
#checkFocus()

**View com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.mEmptyView** `[private]`

View to show if there are no items to show.

**int com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.mLayoutHeight** `[private]`

Our height after the last layout

**SelectionNotifier com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.mSelectionNotifier = null** `[private]`

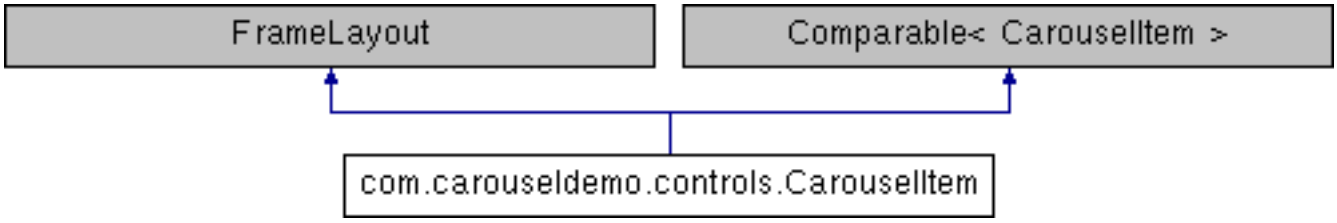
---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/controls/**CarouselAdapter.java**

## com.carouseldemo.controls.CarouselItem Class Reference

Inheritance diagram for com.carouseldemo.controls.CarouselItem:




### Public Member Functions

- **CarouselItem** (Context context, boolean advert)
- String **getColor** ()
- String **getName** ()
- void **setIndex** (int **index**)
- int **getIndex** ()
- String **getItemId** ()
- void **setItemId** (String Id)
- void **setImagePath** (String str)
- String **getImagePath** ()
- void **setCurrentAngle** (float **currentAngle**)
- float **getCurrentAngle** ()
- int **compareTo** (**CarouselItem** another)
- void **setItemX** (float x)
- float **getItemX** ()
- void **setItemY** (float y)
- float **getItemY** ()
- void **setItemZ** (float z)
- float **getItemZ** ()
- void **setDrawn** (boolean **drawn**)
- boolean **isDrawn** ()
- void **setImageBitmap** (Bitmap bitmap)
- Bitmap **getImageBitmap** ()
- void **setText** (String txt)

### Public Attributes

- ImageView **mImage**
- boolean **isAdvert**

### Private Attributes

- int **index**
- float **currentAngle**
- float **itemX**
- float **itemY**
- float **itemZ**
- boolean **drawn**
- String **mImagePath**
- String **id**
- Matrix **mCIMatrix**

## Constructor & Destructor Documentation

**com.carouseldemo.controls.CarouselItem.CarouselItem (Context *context*, boolean *advert*)**

## Member Function Documentation

**int com.carouseldemo.controls.CarouselItem.compareTo (CarouselItem *another*)**
**String com.carouseldemo.controls.CarouselItem.getColor ()**
**float com.carouseldemo.controls.CarouselItem.getCurrentAngle ()**
**Bitmap com.carouseldemo.controls.CarouselItem.getImageBitmap ()**
**String com.carouseldemo.controls.CarouselItem.getImagePath ()**
**int com.carouseldemo.controls.CarouselItem.getIndex ()**
**String com.carouseldemo.controls.CarouselItem.getItemId ()**
**float com.carouseldemo.controls.CarouselItem.getItemX ()**
**float com.carouseldemo.controls.CarouselItem.getItemY ()**
**float com.carouseldemo.controls.CarouselItem.getItemZ ()**
**String com.carouseldemo.controls.CarouselItem.getName ()**
**boolean com.carouseldemo.controls.CarouselItem.isDrawn ()**
**void com.carouseldemo.controls.CarouselItem.setCurrentAngle (float *currentAngle*)**
**void com.carouseldemo.controls.CarouselItem.setDrawn (boolean *drawn*)**
**void com.carouseldemo.controls.CarouselItem.setImageBitmap (Bitmap *bitmap*)**
**void com.carouseldemo.controls.CarouselItem.setImagePath (String *str*)**
**void com.carouseldemo.controls.CarouselItem.setIndex (int *index*)**
**void com.carouseldemo.controls.CarouselItem.setItemId (String *id*)**
**void com.carouseldemo.controls.CarouselItem.setItemX (float *x*)**
**void com.carouseldemo.controls.CarouselItem.setItemY (float *y*)**
**void com.carouseldemo.controls.CarouselItem.setItemZ (float *z*)**
**void com.carouseldemo.controls.CarouselItem.setText (String *txt*)**

## Member Data Documentation

**float com.carouseldemo.controls.CarouselItem.currentAngle** `[private]`
**boolean com.carouseldemo.controls.CarouselItem.drawn** `[private]`
**String com.carouseldemo.controls.CarouselItem.id** `[private]`
**int com.carouseldemo.controls.CarouselItem.index** `[private]`
**boolean com.carouseldemo.controls.CarouselItem.isAdvert**
**float com.carouseldemo.controls.CarouselItem.itemX** `[private]`
**float com.carouseldemo.controls.CarouselItem.itemY** `[private]`
**float com.carouseldemo.controls.CarouselItem.itemZ** `[private]`
**Matrix com.carouseldemo.controls.CarouselItem.mCIMatrix** `[private]`
**ImageView com.carouseldemo.controls.CarouselItem.mImage**
**String com.carouseldemo.controls.CarouselItem.mImagePath** `[private]`

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/controls/**CarouselItem.java**

# com.carouseldemo.controls.CarouselPoint Class Reference

## Public Member Functions

- **CarouselPoint** (float x, float y, float z, float angle)
- float **getX** ()
- float **getY** ()
- float **getZ** ()
- float **getAngleOffset** ()

## Private Attributes

- float **carX**
- float **carY**
- float **carZ**
- float **angleOffset**

---

## Constructor & Destructor Documentation

**com.carouseldemo.controls.CarouselPoint.CarouselPoint (float *x*, float *y*, float *z*, float *angle*)**

---

## Member Function Documentation

**float com.carouseldemo.controls.CarouselPoint.getAngleOffset ()**
**float com.carouseldemo.controls.CarouselPoint.getX ()**
**float com.carouseldemo.controls.CarouselPoint.getY ()**
**float com.carouseldemo.controls.CarouselPoint.getZ ()**

---

## Member Data Documentation

**float com.carouseldemo.controls.CarouselPoint.angleOffset** `[private]`
**float com.carouseldemo.controls.CarouselPoint.carX** `[private]`
**float com.carouseldemo.controls.CarouselPoint.carY** `[private]`
**float com.carouseldemo.controls.CarouselPoint.carZ** `[private]`

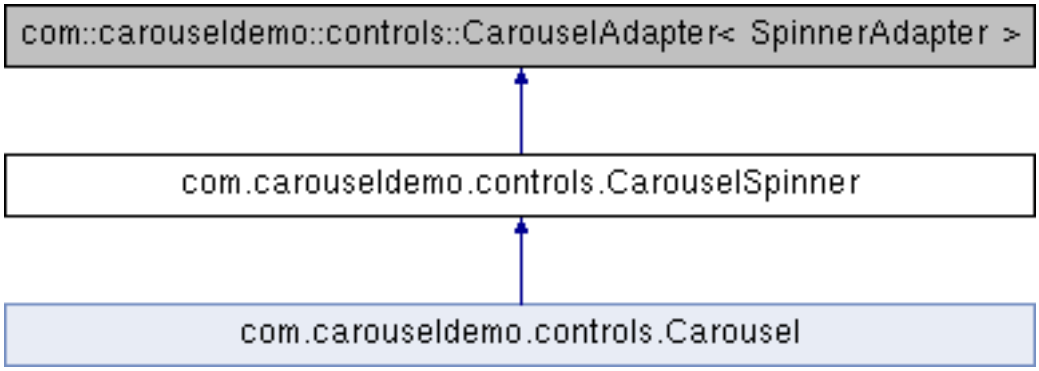
---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/controls/**CarouselPoint.java**

## com.carouseldemo.controls.CarouselSpinner Class Reference

Inheritance diagram for com.carouseldemo.controls.CarouselSpinner:




### Classes

- class **RecycleBin**
- class **SavedState**

### Public Member Functions

- **CarouselSpinner** (Context context)
- **CarouselSpinner** (Context context, AttributeSet attrs)
- **CarouselSpinner** (Context context, AttributeSet attrs, int defStyle)
- SpinnerAdapter **getAdapter** ()
- void **setAdapter** (SpinnerAdapter adapter)
- View **getSelectedView** ()
- void **setSelection** (int position, boolean animate)
- void **setSelection** (int position)
- void **requestLayout** ()
- int **pointToPosition** (int x, int y)
- Parcelable **onSaveInstanceState** ()
- void **onRestoreInstanceState** (Parcelable state)

### Public Attributes

- SpinnerAdapter **mAdapter**
- final Rect **mSpinnerPadding** = new Rect()
- final RecycleBin **mRecycler** = new RecycleBin()

### Protected Member Functions

- void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec)
- ViewGroup.LayoutParams **generateDefaultLayoutParams** ()

### Private Member Functions

- void **initCarouselSpinner** ()

### Private Attributes

- boolean **mBlockLayoutRequests**
- int **mSelectionLeftPadding** = 0
- int **mSelectionTopPadding** = 0
- int **mSelectionRightPadding** = 0

·int **mSelectionBottomPadding** = 0
·DataSetObserver **mDataSetObserver**

---

## Constructor & Destructor Documentation

**com.carouseldemo.controls.CarouselSpinner.CarouselSpinner (Context *context*)**
**com.carouseldemo.controls.CarouselSpinner.CarouselSpinner (Context *context*, AttributeSet *attrs*)**
**com.carouseldemo.controls.CarouselSpinner.CarouselSpinner (Context *context*, AttributeSet *attrs*, int *defStyle*)**

---

## Member Function Documentation

**ViewGroup.LayoutParams com.carouseldemo.controls.CarouselSpinner.generateDefaultLayoutParams () `[protected]`**
**SpinnerAdapter com.carouseldemo.controls.CarouselSpinner.getAdapter ()**
**View com.carouseldemo.controls.CarouselSpinner.getSelectedView ()**
**void com.carouseldemo.controls.CarouselSpinner.initCarouselSpinner () `[private]`**

Common code for different constructor flavors

**void com.carouseldemo.controls.CarouselSpinner.onMeasure (int *widthMeasureSpec*, int *heightMeasureSpec*) `[protected]`**

**See Also:**

android.view.View::measure(int, int)

Figure out the dimensions of this Spinner. The width comes from the widthMeasureSpec as Spinners can't have their width set to UNSPECIFIED. The height is based on the height of the selected item plus padding.

**void com.carouseldemo.controls.CarouselSpinner.onRestoreInstanceState (Parcelable *state*)**
**Parcelable com.carouseldemo.controls.CarouselSpinner.onSaveInstanceState ()**
**int com.carouseldemo.controls.CarouselSpinner.pointToPosition (int *x*, int *y*)**

Maps a point to a position in the list.

**Parameters:**

| | |
|---|---|
| *x* | X in local coordinate |
| *y* | Y in local coordinate |

**Returns:**

The position of the item which contains the specified point, or **INVALID_POSITION** if the point does not intersect an item.

**void com.carouseldemo.controls.CarouselSpinner.requestLayout ()**

Override to prevent spamming ourselves with layout requests as we place views

**See Also:**

android.view.View::requestLayout()

**void com.carouseldemo.controls.CarouselSpinner.setAdapter (SpinnerAdapter *adapter*)**
**void com.carouseldemo.controls.CarouselSpinner.setSelection (int *position*, boolean *animate*)**

Jump directly to a specific item in the adapter data.

**void com.carouseldemo.controls.CarouselSpinner.setSelection (int *position*)**

## Member Data Documentation

**SpinnerAdapter com.carouseldemo.controls.CarouselSpinner.mAdapter**
**boolean com.carouseldemo.controls.CarouselSpinner.mBlockLayoutRequests** `[private]`
**DataSetObserver com.carouseldemo.controls.CarouselSpinner.mDataSetObserver** `[private]`
**final RecycleBin com.carouseldemo.controls.CarouselSpinner.mRecycler = new RecycleBin()**
**int com.carouseldemo.controls.CarouselSpinner.mSelectionBottomPadding = 0** `[private]`
**int com.carouseldemo.controls.CarouselSpinner.mSelectionLeftPadding = 0** `[private]`
**int com.carouseldemo.controls.CarouselSpinner.mSelectionRightPadding = 0** `[private]`
**int com.carouseldemo.controls.CarouselSpinner.mSelectionTopPadding = 0** `[private]`
**final Rect com.carouseldemo.controls.CarouselSpinner.mSpinnerPadding = new Rect()**

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/controls/**CarouselSpinner.java**

# com.carouseldemo.controls.Command Class Reference

## Public Attributes

- String **name**
- HashMap< String, String > **attributes**
- boolean **isRepeatBlock**
- **Commands repeatCommands**

---

## Member Data Documentation

**HashMap<String, String> com.carouseldemo.controls.Command.attributes**
**boolean com.carouseldemo.controls.Command.isRepeatBlock**
**String com.carouseldemo.controls.Command.name**
**Commands com.carouseldemo.controls.Command.repeatCommands**

---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/controls/**Command.java**

# com.carouseldemo.controls.Commands Class Reference

## Public Member Functions

- boolean **addCommand** (**Command** c)
- boolean **removeCommand** (**Command** c)
- **Command removeCommand** (int i)
- int **getIndex** (**Command** c)
- **Command getNextCommand** ()

## Private Member Functions

- void **updatePrevCommands** ()

---

## Member Function Documentation

**boolean com.carouseldemo.controls.Commands.addCommand (Command *c*)**
**int com.carouseldemo.controls.Commands.getIndex (Command *c*)**
**Command com.carouseldemo.controls.Commands.getNextCommand ()**
**boolean com.carouseldemo.controls.Commands.removeCommand (Command *c*)**
**Command com.carouseldemo.controls.Commands.removeCommand (int *i*)**
**void com.carouseldemo.controls.Commands.updatePrevCommands () `[private]`**

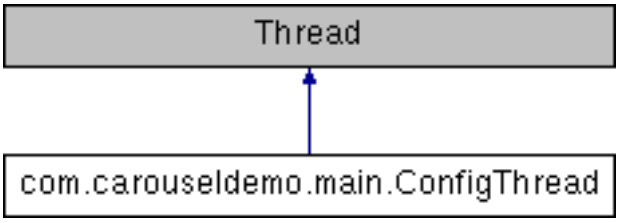
---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/controls/**Commands.java**

# com.carouseldemo.main.ConfigThread Class Reference

Inheritance diagram for com.carouseldemo.main.ConfigThread:




## Public Member Functions

- **ConfigThread** (Context ctx)
- void **pleaseStop** ()
- void **run** ()
- void **pleasePause** ()
- void **pleaseResume** ()

## Private Member Functions

- boolean **confirmDownload** (String file)

## Private Attributes

- final String **getConfigScript** = "scripts/cnfms/getConfig.php?terminal="
- final String **confirmDownloadScript** = "scripts/cnfms/ackConfigFile.php?terminal="
- boolean **shouldWork** = true
- Object **mPauseLock** = new Object()
- boolean **mPaused** = false
- Context **context** = null

## Static Private Attributes

- static final String **TAG** = ConfigThread.class.getSimpleName()

---

## Constructor & Destructor Documentation

**com.carouseldemo.main.ConfigThread.ConfigThread (Context *ctx*)**

---

## Member Function Documentation

**boolean com.carouseldemo.main.ConfigThread.confirmDownload (String *file*)** `[private]`

**void com.carouseldemo.main.ConfigThread.pleasePause ()**

Call this on pause.

**void com.carouseldemo.main.ConfigThread.pleaseResume ()**

Call this on resume.

**void com.carouseldemo.main.ConfigThread.pleaseStop ()**

**void com.carouseldemo.main.ConfigThread.run ()**

---

## Member Data Documentation

**final String com.carouseldemo.main.ConfigThread.confirmDownloadScript = "scripts/cnfms/ackConfigFile.php?terminal="** `[private]`

**Context com.carouseldemo.main.ConfigThread.context = null** `[private]`

**final String com.carouseldemo.main.ConfigThread.getConfigScript = "scripts/cnfms/getConfig.php?terminal="** `[private]`

**boolean com.carouseldemo.main.ConfigThread.mPaused = false** `[private]`

**Object com.carouseldemo.main.ConfigThread.mPauseLock = new Object()** `[private]`

**boolean com.carouseldemo.main.ConfigThread.shouldWork = true** `[private]`

**final String com.carouseldemo.main.ConfigThread.TAG = ConfigThread.class.getSimpleName()** `[static], [private]`

Tag for a class logging

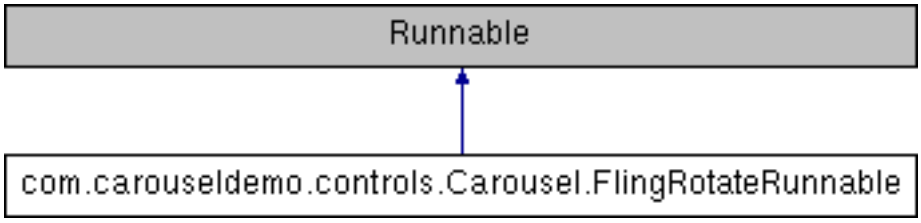
---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/main/**ConfigThread.java**

# com.carouseldemo.controls.Carousel.FlingRotateRunnable Class Reference

Inheritance diagram for com.carouseldemo.controls.Carousel.FlingRotateRunnable:




## Public Member Functions

- **FlingRotateRunnable** ()
- void **startUsingVelocity** (float initialVelocity)
- void **startUsingDistance** (float deltaAngle)
- void **stop** (boolean **scrollIntoSlots**)
- void **run** ()

## Private Member Functions

- void **startCommon** ()
- void **endFling** (boolean **scrollIntoSlots**)

## Private Attributes

- **Rotator mRotator**
- float **mLastFlingAngle**

---

## Constructor & Destructor Documentation

**com.carouseldemo.controls.Carousel.FlingRotateRunnable.FlingRotateRunnable ()**

Constructor

---

## Member Function Documentation

**void com.carouseldemo.controls.Carousel.FlingRotateRunnable.endFling (boolean *scrollIntoSlots*) `[private]`**
**void com.carouseldemo.controls.Carousel.FlingRotateRunnable.run ()**
**void com.carouseldemo.controls.Carousel.FlingRotateRunnable.startCommon () `[private]`**
**void com.carouseldemo.controls.Carousel.FlingRotateRunnable.startUsingDistance (float *deltaAngle*)**
**void com.carouseldemo.controls.Carousel.FlingRotateRunnable.startUsingVelocity (float *initialVelocity*)**
**void com.carouseldemo.controls.Carousel.FlingRotateRunnable.stop (boolean *scrollIntoSlots*)**

---

## Member Data Documentation

**float com.carouseldemo.controls.Carousel.FlingRotateRunnable.mLastFlingAngle `[private]`**

Angle value reported by mRotator on the previous fling

**Rotator com.carouseldemo.controls.Carousel.FlingRotateRunnable.mRotator `[private]`**

Tracks the decay of a fling rotation

---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/controls/**Carousel.java**

# com.carouseldemo.main.Helper Class Reference

## Classes

·class **ButtonTimerTask**

## Static Public Member Functions

·static Bitmap **QR_Encode** (String data, int h, int w)
·static Bundle **getOfferProperties** (String estateId)
·static boolean **deleteRecursive** (File dir)
·static boolean **createOfferThumbnail** (Context ctx, File offerDir, String id, JSONObject details)
·static Bitmap **createAdvertThumbnail** (Context ctx, File thumb)
·static boolean **writeDomDocumentToFile** (Node dom, String fileName)
·static boolean **createOfferProperties** (File offerDir, String id, JSONObject details)
·static String **getDeviceId** (Context context)
·static String **getActiveFilterSet** ()
·static int **getRequestInterval** ()
·static String **getFilterByName** (String filterset)
·static String **getActiveCommandSet** ()
·static **Commands getCommandSetByName** (String commandSetToFind)
·static int **getContextCount** ()
·static boolean **performCommand** (int command)
·static Hashtable< Float,
·**CarouselPoint** > **getTrajectory** (int width, String pathName)
·static String **getTrajectoryName** ()
·static int **interpretCount** (String count, int contextCount)
·static Activity **getCurrentActivity** ()
·static void **fillInOfferProps** ()
·static void **startActivityTimer** ()
·static void **cancelActivityTimer** ()
·static boolean **downloadImageTo** (String name, File offerDir, String front, String type)
·static boolean **downloadConfigFileTo** (String name, File carouselDir)
·static void **deleteHistoryEntry** (String id)
·static void **deleteHistory** ()
·static boolean **checkForExternalStorage** (int seconds)

## Static Public Attributes

·static final int **WHITE** = 0x00FFFFFF
·static final int **BLACK** = 0xFF000000
·static final int **COMMAND_TURN_LEFT** = 0
·static final int **COMMAND_TURN_RIGHT** = 1
·static final int **COMMAND_ENTER** = 2
·static final int **COMMAND_GO_BACK** = 3
·static final int **COMMAND_GO_HOME** = 4
·static final int **COMMAND_UPDATE** = 5
·static final int **LAST_POSITION_CAROUSEL** = 0
·static final int **LAST_POSITION_ITEM** = 1
·static int **LastPosition** = **LAST_POSITION_CAROUSEL**
·static ButtonTimerTask **buttonTimerTask** = new ButtonTimerTask()
·static Timer **buttonTimer** = new Timer()
·static HashMap< String, Bundle > **offerProperties** = new HashMap<String, Bundle>()
·static boolean **isManualControl** = false
·static boolean **isDownloadInterrupted** = false

- static boolean **dialogStarted** = false
- static final String **server** = "http://212.204.78.26:3000/"
- static final String **AppTAG** = "CarouselInfo"

## Static Protected Attributes

- static long **userActivityTimeout** = 30000

## Static Private Member Functions

- static **Commands retrieveCommands** (Element commandSet)

## Static Private Attributes

- static int[] **colors** = new int[210*210]
- static final String **TAG** = Helper.class.getSimpleName()

---

## Member Function Documentation

**static void com.carouseldemo.main.Helper.cancelActivityTimer () `[static]`**
**static boolean com.carouseldemo.main.Helper.checkForExternalStorage (int *seconds*) `[static]`**
**static Bitmap com.carouseldemo.main.Helper.createAdvertThumbnail (Context *ctx*, File *thumb*) `[static]`**
**static boolean com.carouseldemo.main.Helper.createOfferProperties (File *offerDir*, String *id*, JSONObject *details*) `[static]`**
**static boolean com.carouseldemo.main.Helper.createOfferThumbnail (Context *ctx*, File *offerDir*, String *id*, JSONObject *details*) `[static]`**
**static void com.carouseldemo.main.Helper.deleteHistory () `[static]`**
**static void com.carouseldemo.main.Helper.deleteHistoryEntry (String *id*) `[static]`**
**static boolean com.carouseldemo.main.Helper.deleteRecursive (File *dir*) `[static]`**
**static boolean com.carouseldemo.main.Helper.downloadConfigFileTo (String *name*, File *carouselDir*) `[static]`**
**static boolean com.carouseldemo.main.Helper.downloadImageTo (String *name*, File *offerDir*, String *front*, String *type*) `[static]`**
**static void com.carouseldemo.main.Helper.fillInOfferProps () `[static]`**
**static String com.carouseldemo.main.Helper.getActiveCommandSet () `[static]`**
**static String com.carouseldemo.main.Helper.getActiveFilterSet () `[static]`**
**static Commands com.carouseldemo.main.Helper.getCommandSetByName (String *commandSetToFind*) `[static]`**
**static int com.carouseldemo.main.Helper.getContextCount () `[static]`**
**static Activity com.carouseldemo.main.Helper.getCurrentActivity () `[static]`**
**static String com.carouseldemo.main.Helper.getDeviceId (Context *context*) `[static]`**
**static String com.carouseldemo.main.Helper.getFilterByName (String *filterset*) `[static]`**
**static Bundle com.carouseldemo.main.Helper.getOfferProperties (String *estateId*) `[static]`**
**static int com.carouseldemo.main.Helper.getRequestInterval () `[static]`**
**static Hashtable<Float, CarouselPoint> com.carouseldemo.main.Helper.getTrajectory (int *width*, String *pathName*) `[static]`**
**static String com.carouseldemo.main.Helper.getTrajectoryName () `[static]`**
**static int com.carouseldemo.main.Helper.interpretCount (String *count*, int *contextCount*) `[static]`**
**static boolean com.carouseldemo.main.Helper.performCommand (int *command*) `[static]`**
**static Bitmap com.carouseldemo.main.Helper.QR_Encode (String *data*, int *h*, int *w*) `[static]`**
**static Commands com.carouseldemo.main.Helper.retrieveCommands (Element *commandSet*) `[static]`, `[private]`**

**static void com.carouseldemo.main.Helper.startActivityTimer ()** `[static]`
**static boolean com.carouseldemo.main.Helper.writeDomDocumentToFile (Node *dom*, String *fileName*)** `[static]`

---

## Member Data Documentation

**final String com.carouseldemo.main.Helper.AppTAG = "CarouselInfo"** `[static]`
**final int com.carouseldemo.main.Helper.BLACK = 0xFF000000** `[static]`
**Timer com.carouseldemo.main.Helper.buttonTimer = new Timer()** `[static]`
**ButtonTimerTask com.carouseldemo.main.Helper.buttonTimerTask = new ButtonTimerTask()** `[static]`
**int [] com.carouseldemo.main.Helper.colors = new int[210*210]** `[static], [private]`
**final int com.carouseldemo.main.Helper.COMMAND_ENTER = 2** `[static]`
**final int com.carouseldemo.main.Helper.COMMAND_GO_BACK = 3** `[static]`
**final int com.carouseldemo.main.Helper.COMMAND_GO_HOME = 4** `[static]`
**final int com.carouseldemo.main.Helper.COMMAND_TURN_LEFT = 0** `[static]`
**final int com.carouseldemo.main.Helper.COMMAND_TURN_RIGHT = 1** `[static]`
**final int com.carouseldemo.main.Helper.COMMAND_UPDATE = 5** `[static]`
**boolean com.carouseldemo.main.Helper.dialogStarted = false** `[static]`
**boolean com.carouseldemo.main.Helper.isDownloadInterrupted = false** `[static]`
**boolean com.carouseldemo.main.Helper.isManualControl = false** `[static]`
**final int com.carouseldemo.main.Helper.LAST_POSITION_CAROUSEL = 0** `[static]`
**final int com.carouseldemo.main.Helper.LAST_POSITION_ITEM = 1** `[static]`
**int com.carouseldemo.main.Helper.LastPosition = LAST_POSITION_CAROUSEL** `[static]`
**HashMap<String, Bundle> com.carouseldemo.main.Helper.offerProperties = new HashMap<String, Bundle>()** `[static]`
**final String com.carouseldemo.main.Helper.server = "http://212.204.78.26:3000/"** `[static]`
**final String com.carouseldemo.main.Helper.TAG = Helper.class.getSimpleName()** `[static], [private]`

Tag for a class logging

**long com.carouseldemo.main.Helper.userActivityTimeout = 30000** `[static], [protected]`
**final int com.carouseldemo.main.Helper.WHITE = 0x00FFFFFF** `[static]`

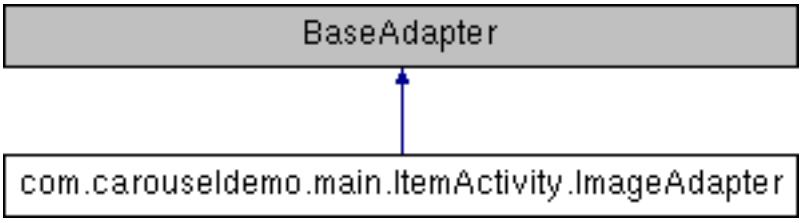
---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/main/**Helper.java**

# com.carouseldemo.main.ItemActivity.ImageAdapter Class Reference

Inheritance diagram for com.carouseldemo.main.ItemActivity.ImageAdapter:




## Public Member Functions

- **ImageAdapter** (Context c)
- int **getCount** ()
- Object **getItem** (int position)
- long **getItemId** (int position)
- View **getView** (int position, View convertView, ViewGroup parent)

## Private Attributes

- Context **mContext**

---

## Constructor & Destructor Documentation

**com.carouseldemo.main.ItemActivity.ImageAdapter.ImageAdapter (Context *c*)**

---

## Member Function Documentation

**int com.carouseldemo.main.ItemActivity.ImageAdapter.getCount ()**
**Object com.carouseldemo.main.ItemActivity.ImageAdapter.getItem (int *position*)**
**long com.carouseldemo.main.ItemActivity.ImageAdapter.getItemId (int *position*)**
**View com.carouseldemo.main.ItemActivity.ImageAdapter.getView (int *position*, View *convertView*, ViewGroup *parent*)**

---

## Member Data Documentation

**Context com.carouseldemo.main.ItemActivity.ImageAdapter.mContext** `[private]`

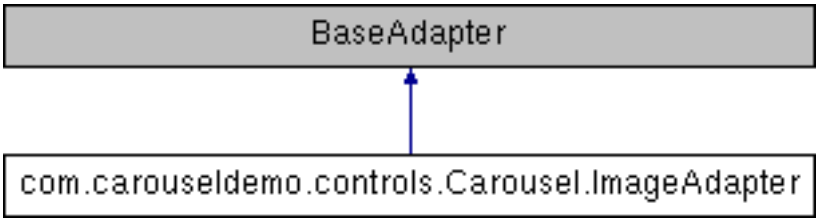
---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/main/**ItemActivity.java**

# com.carouseldemo.controls.Carousel.ImageAdapter Class Reference

Inheritance diagram for com.carouseldemo.controls.Carousel.ImageAdapter:




## Public Member Functions

- **ImageAdapter** (Context c)
- String **getImagePathFromId** (int id)
- void **readImages** ()
- void **SetImages** (TypedArray array, TypedArray names)
- void **SetImages** (TypedArray array, TypedArray names, boolean bReflected)
- int **getCount** ()
- Object **getItem** (int position)
- long **getItemId** (int position)
- View **getView** (int position, View convertView, ViewGroup parent)
- void **removeItem** (int id)
- void **addItem** (File f)
- void **changeItemContent** (int id, File f)

## Public Attributes

- **CarouselItem**[] **mImages**
- Bitmap[] **itemThumb**
- Bitmap **videoThumb** = null
- int **imageCount** = 0

## Private Attributes

- Context **mContext**
- ArrayList< File > **images** = null
- ArrayList< String > **estateIds** = null
- final int **MAX_IMAGE_COUNT** = 20

---

## Constructor & Destructor Documentation

**com.carouseldemo.controls.Carousel.ImageAdapter.ImageAdapter (Context *c*)**

---

## Member Function Documentation

**void com.carouseldemo.controls.Carousel.ImageAdapter.addItem (File *f*)**
**void com.carouseldemo.controls.Carousel.ImageAdapter.changeItemContent (int *id*, File *f*)**
**int com.carouseldemo.controls.Carousel.ImageAdapter.getCount ()**
**String com.carouseldemo.controls.Carousel.ImageAdapter.getImagePathFromId (int *id*)**
**Object com.carouseldemo.controls.Carousel.ImageAdapter.getItem (int *position*)**

**long com.carouseldemo.controls.Carousel.ImageAdapter.getItemId (int *position*)**
**View com.carouseldemo.controls.Carousel.ImageAdapter.getView (int *position*, View *convertView*, ViewGroup *parent*)**
**void com.carouseldemo.controls.Carousel.ImageAdapter.readImages ()**
**void com.carouseldemo.controls.Carousel.ImageAdapter.removeItem (int *id*)**
**void com.carouseldemo.controls.Carousel.ImageAdapter.SetImages (TypedArray *array*, TypedArray *names*)**
**void com.carouseldemo.controls.Carousel.ImageAdapter.SetImages (TypedArray *array*, TypedArray *names*, boolean *bReflected*)**

---

## Member Data Documentation

**ArrayList<String> com.carouseldemo.controls.Carousel.ImageAdapter.estateIds = null** `[private]`
**int com.carouseldemo.controls.Carousel.ImageAdapter.imageCount = 0**
**ArrayList<File> com.carouseldemo.controls.Carousel.ImageAdapter.images = null** `[private]`
**Bitmap [] com.carouseldemo.controls.Carousel.ImageAdapter.itemThumb**
**final int com.carouseldemo.controls.Carousel.ImageAdapter.MAX_IMAGE_COUNT = 20** `[private]`
**Context com.carouseldemo.controls.Carousel.ImageAdapter.mContext** `[private]`
**CarouselItem [] com.carouseldemo.controls.Carousel.ImageAdapter.mImages**
**Bitmap com.carouseldemo.controls.Carousel.ImageAdapter.videoThumb = null**

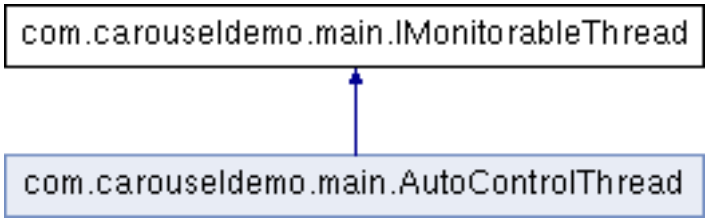
---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/controls/**Carousel.java**

## com.carouseldemo.main.IMonitorableThread Interface Reference

Inheritance diagram for com.carouseldemo.main.IMonitorableThread:




### Public Member Functions

- boolean **isPaused** ()
- void **setThreadMonitor** (**IThreadMonitor** mon)

---

### Member Function Documentation

**boolean com.carouseldemo.main.IMonitorableThread.isPaused ()**

Implemented in **com.carouseldemo.main.AutoControlThread** (*p.*).

**void com.carouseldemo.main.IMonitorableThread.setThreadMonitor (IThreadMonitor *mon*)**

Implemented in **com.carouseldemo.main.AutoControlThread** (*p.*).

---

**The documentation for this interface was generated from the following file:**

- src/com/carouseldemo/main/**IMonitorableThread.java**

# com.carouseldemo.main.ItemActivity Class Reference

Inheritance diagram for com.carouseldemo.main.ItemActivity:


## Classes

- class **ImageAdapter**
- class **PropsTask**

## Public Member Functions

- boolean **onKeyUp** (int keyCode, KeyEvent event)
- void **onBackPressed** ()
- void **onCreate** (Bundle savedInstanceState)
- boolean **playVideo** ()
- void **onItemSelected** (AdapterView<?> parent, View v, int position, long id)
- void **onNothingSelected** (AdapterView<?> parent)

## Static Public Member Functions

- static void **performCommand** (int command)

## Public Attributes

- Gallery **g** = null

## Static Public Attributes

- static **ItemActivity mActivity** = null
- static int **itemCount** = 0
- static Animation **fadeIn**
- static Animation **fadeOut**

## Private Member Functions

- void **goToImageView** ()
- void **goToVideoView** ()
- void **drawBigView** (int position)
- void **initMediaFileList** ()

## Static Private Member Functions

- static void **animateTransition** ()
- static void **updatePositionToRight** ()
- static void **updatePositionToLeft** ()

## Private Attributes

- ArrayList< String > **mMediaFilesBig** = new ArrayList<String>()
- ArrayList< String > **mMediaFilesSmall** = new ArrayList<String>()

·String **estateId** = null
·ImageView **mImageView** = null
·VideoView **mVideoView** = null
·View **lastView** = null
·ImageView **mQRImage** = null
·boolean **videoPlaying** = false
·boolean **backPressed** = false
·Drawable **background** = null
·Bitmap **bm**
·Bitmap **b**
·Bitmap **videoThumb**
·Bitmap **thumb**
·Bitmap **nav_bm** = null
·ImageView **navImage**
·ImageView **i**
·LinearLayout **itemLayout** = null
·Hashtable< String, Bitmap > **galleryBin** = new Hashtable<String, Bitmap>(150)
·long **freeSize** = 0L
·long **totalSize** = 0L
·long **usedSize** = -1L

## Static Private Attributes

·static int[] **colors** = null
·static Bitmap **blackTrans** = null
·static int **positionTo** = 0
·static Bitmap **bmp**
·static final String **TAG** = ItemActivity.class.getSimpleName()

---

## Member Function Documentation

**static void com.carouseldemo.main.ItemActivity.animateTransition ()** `[static], [private]`
**void com.carouseldemo.main.ItemActivity.drawBigView (int *position*)** `[private]`
**void com.carouseldemo.main.ItemActivity.goToImageView ()** `[private]`
**void com.carouseldemo.main.ItemActivity.goToVideoView ()** `[private]`
**void com.carouseldemo.main.ItemActivity.initMediaFileList ()** `[private]`
**void com.carouseldemo.main.ItemActivity.onBackPressed ()**
**void com.carouseldemo.main.ItemActivity.onCreate (Bundle *savedInstanceState*)**
**void com.carouseldemo.main.ItemActivity.onItemSelected (AdapterView<?> *parent*, View *v*, int *position*, long *id*)**
**boolean com.carouseldemo.main.ItemActivity.onKeyUp (int *keyCode*, KeyEvent *event*)**
**void com.carouseldemo.main.ItemActivity.onNothingSelected (AdapterView<?> *parent*)**
**static void com.carouseldemo.main.ItemActivity.performCommand (int *command*)** `[static]`
**boolean com.carouseldemo.main.ItemActivity.playVideo ()**
**static void com.carouseldemo.main.ItemActivity.updatePositionToLeft ()** `[static], [private]`
**static void com.carouseldemo.main.ItemActivity.updatePositionToRight ()** `[static], [private]`

---

## Member Data Documentation

**Bitmap com.carouseldemo.main.ItemActivity.b** `[private]`

---

**Drawable com.carouseldemo.main.ItemActivity.background = null** `[private]`
**boolean com.carouseldemo.main.ItemActivity.backPressed = false** `[private]`
**Bitmap com.carouseldemo.main.ItemActivity.blackTrans = null** `[static]`, `[private]`
**Bitmap com.carouseldemo.main.ItemActivity.bm** `[private]`
**Bitmap com.carouseldemo.main.ItemActivity.bmp** `[static]`, `[private]`
**int [] com.carouseldemo.main.ItemActivity.colors = null** `[static]`, `[private]`
**String com.carouseldemo.main.ItemActivity.estateId = null** `[private]`
**Animation com.carouseldemo.main.ItemActivity.fadeIn** `[static]`
**Animation com.carouseldemo.main.ItemActivity.fadeOut** `[static]`
**long com.carouseldemo.main.ItemActivity.freeSize = 0L** `[private]`
**Gallery com.carouseldemo.main.ItemActivity.g = null**
**Hashtable<String, Bitmap> com.carouseldemo.main.ItemActivity.galleryBin = new Hashtable<String, Bitmap>(150)** `[private]`
**ImageView com.carouseldemo.main.ItemActivity.i** `[private]`
**int com.carouseldemo.main.ItemActivity.itemCount = 0** `[static]`
**LinearLayout com.carouseldemo.main.ItemActivity.itemLayout = null** `[private]`
**View com.carouseldemo.main.ItemActivity.lastView = null** `[private]`
**ItemActivity com.carouseldemo.main.ItemActivity.mActivity = null** `[static]`
**ImageView com.carouseldemo.main.ItemActivity.mImageView = null** `[private]`
**ArrayList<String> com.carouseldemo.main.ItemActivity.mMediaFilesBig = new ArrayList<String>()** `[private]`
**ArrayList<String> com.carouseldemo.main.ItemActivity.mMediaFilesSmall = new ArrayList<String>()** `[private]`
**ImageView com.carouseldemo.main.ItemActivity.mQRImage = null** `[private]`
**VideoView com.carouseldemo.main.ItemActivity.mVideoView = null** `[private]`
**Bitmap com.carouseldemo.main.ItemActivity.nav_bm = null** `[private]`
**ImageView com.carouseldemo.main.ItemActivity.navImage** `[private]`
**int com.carouseldemo.main.ItemActivity.positionTo = 0** `[static]`, `[private]`
**final String com.carouseldemo.main.ItemActivity.TAG = ItemActivity.class.getSimpleName()** `[static]`, `[private]`

Tag for a class logging

**Bitmap com.carouseldemo.main.ItemActivity.thumb** `[private]`
**long com.carouseldemo.main.ItemActivity.totalSize = 0L** `[private]`
**long com.carouseldemo.main.ItemActivity.usedSize = -1L** `[private]`
**boolean com.carouseldemo.main.ItemActivity.videoPlaying = false** `[private]`
**Bitmap com.carouseldemo.main.ItemActivity.videoThumb** `[private]`

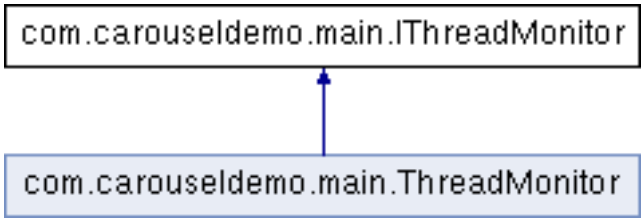
---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/main/**ItemActivity.java**

## com.carouseldemo.main.IThreadMonitor Interface Reference

Inheritance diagram for com.carouseldemo.main.IThreadMonitor:




### Public Member Functions

- boolean **setMonitoredReady** ()
- boolean **bark** ()

---

### Member Function Documentation

**boolean com.carouseldemo.main.IThreadMonitor.bark ()**

Implemented in **com.carouseldemo.main.ThreadMonitor** (*p.*).

**boolean com.carouseldemo.main.IThreadMonitor.setMonitoredReady ()**

Implemented in **com.carouseldemo.main.ThreadMonitor** (*p.*).

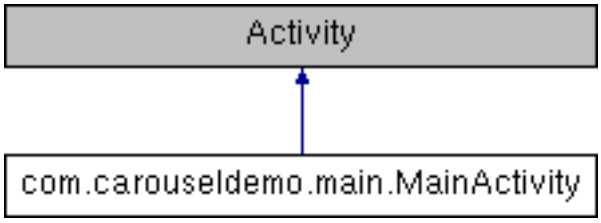
---

**The documentation for this interface was generated from the following file:**

- src/com/carouseldemo/main/**IThreadMonitor.java**

# com.carouseldemo.main.MainActivity Class Reference

Inheritance diagram for com.carouseldemo.main.MainActivity:




## Public Member Functions

- void **onDestroy** ()
- void **onBackPressed** ()
- void **destroyActivity** ()
- boolean **onKeyUp** (int keyCode, KeyEvent event)
- void **onCreate** (Bundle savedInstanceState)

## Static Public Member Functions

- static void **performCommand** (int command)

## Public Attributes

- **Carousel carousel** = null
- **ConfigThread configThread** = null

## Static Public Attributes

- static boolean **created** = false
- static **MainActivity mActivity** = null
- static int **selectedPos** = 0
- static boolean **isFirstStart** = false
- static **AutoControlThread autoControlThread** = null
- static int **itemCount** = 0

## Protected Member Functions

- void **onActivityResult** (int requestCode, int resultCode, Intent data)

## Private Member Functions

- void **cancelAlmostEverything** ()

## Static Private Member Functions

- static void **updatePositionToLeft** ()
- static void **updatePositionToRight** ()

## Static Private Attributes

- static int **positionTo** = 0
- static final int **MONITORING_INTERVAL_MS** = 1000000
- static final String **TAG** = Carousel.class.getSimpleName()

## Member Function Documentation

**void com.carouseldemo.main.MainActivity.cancelAlmostEverything () `[private]`**
**void com.carouseldemo.main.MainActivity.destroyActivity ()**
**void com.carouseldemo.main.MainActivity.onActivityResult (int *requestCode*, int *resultCode*, Intent *data*) `[protected]`**
**void com.carouseldemo.main.MainActivity.onBackPressed ()**
**void com.carouseldemo.main.MainActivity.onCreate (Bundle *savedInstanceState*)**
**void com.carouseldemo.main.MainActivity.onDestroy ()**
**boolean com.carouseldemo.main.MainActivity.onKeyUp (int *keyCode*, KeyEvent *event*)**
**static void com.carouseldemo.main.MainActivity.performCommand (int *command*) `[static]`**
**static void com.carouseldemo.main.MainActivity.updatePositionToLeft () `[static]`, `[private]`**
**static void com.carouseldemo.main.MainActivity.updatePositionToRight () `[static]`, `[private]`**

---

## Member Data Documentation

**AutoControlThread com.carouseldemo.main.MainActivity.autoControlThread = null `[static]`**
**Carousel com.carouseldemo.main.MainActivity.carousel = null**
**ConfigThread com.carouseldemo.main.MainActivity.configThread = null**
**boolean com.carouseldemo.main.MainActivity.created = false `[static]`**

Called when the activity is first created.

**boolean com.carouseldemo.main.MainActivity.isFirstStart = false `[static]`**
**int com.carouseldemo.main.MainActivity.itemCount = 0 `[static]`**
**MainActivity com.carouseldemo.main.MainActivity.mActivity = null `[static]`**
**final int com.carouseldemo.main.MainActivity.MONITORING_INTERVAL_MS = 1000000 `[static]`, `[private]`**
**int com.carouseldemo.main.MainActivity.positionTo = 0 `[static]`, `[private]`**
**int com.carouseldemo.main.MainActivity.selectedPos = 0 `[static]`**
**final String com.carouseldemo.main.MainActivity.TAG = Carousel.class.getSimpleName() `[static]`, `[private]`**

Tag for a class logging

---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/main/**MainActivity.java**

# com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.OnItemClickListener Interface Reference

## Public Member Functions

· void **onItemClick** (**CarouselAdapter**<?> parent, View view, int position, long id)

---

## Detailed Description

Interface definition for a callback to be invoked when an item in this CarouselAdapter has been clicked.

---

## Member Function Documentation

### void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.OnItemClickListener.onItemClick (CarouselAdapter<?> *parent*, View *view*, int *position*, long *id*)

Callback method to be invoked when an item in this CarouselAdapter has been clicked.

Implementers can call getItemAtPosition(position) if they need to access the data associated with the selected item.

**Parameters:**

| | |
|---|---|
| *parent* | The CarouselAdapter where the click happened. |
| *view* | The view within the CarouselAdapter that was clicked (this will be a view provided by the adapter) |
| *position* | The position of the view in the adapter. |
| *id* | The row id of the item that was clicked. |

---

**The documentation for this interface was generated from the following file:**

· src/com/carouseldemo/controls/**CarouselAdapter.java**

# com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.OnItemLongClickListener Interface Reference

## Public Member Functions

·boolean **onItemLongClick** (**CarouselAdapter**<?> parent, View view, int position, long id)

---

## Detailed Description

Interface definition for a callback to be invoked when an item in this view has been clicked and held.

---

## Member Function Documentation

### boolean com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.OnItemLongClickListener.onItemLongClick (CarouselAdapter<?> *parent*, View *view*, int *position*, long *id*)

Callback method to be invoked when an item in this view has been clicked and held.

Implementers can call getItemAtPosition(position) if they need to access the data associated with the selected item.

**Parameters:**

| | |
|---|---|
| *parent* | The AbsListView where the click happened |
| *view* | The view within the AbsListView that was clicked |
| *position* | The position of the view in the list |
| *id* | The row id of the item that was clicked |

**Returns:**

true if the callback consumed the long click, false otherwise

---

**The documentation for this interface was generated from the following file:**

·src/com/carouseldemo/controls/**CarouselAdapter.java**

# com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.OnItemSelectedListener Interface Reference

## Public Member Functions

- void **onItemSelected** (**CarouselAdapter**<?> parent, View view, int position, long id)
- void **onNothingSelected** (**CarouselAdapter**<?> parent)

---

## Detailed Description

Interface definition for a callback to be invoked when an item in this view has been selected.

---

## Member Function Documentation

### void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.OnItemSelectedListener.onItemSelected (CarouselAdapter<?> *parent*, View *view*, int *position*, long *id*)

Callback method to be invoked when an item in this view has been selected.

Impelmenters can call getItemAtPosition(position) if they need to access the data associated with the selected item.

**Parameters:**

| | |
|---|---|
| *parent* | The CarouselAdapter where the selection happened |
| *view* | The view within the CarouselAdapter that was clicked |
| *position* | The position of the view in the adapter |
| *id* | The row id of the item that is selected |

### void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.OnItemSelectedListener.onNothingSelected (CarouselAdapter<?> *parent*)

Callback method to be invoked when the selection disappears from this view. The selection can disappear for instance when touch is activated or when the adapter becomes empty.

**Parameters:**

| | |
|---|---|
| *parent* | The CarouselAdapter that now contains no selected item. |

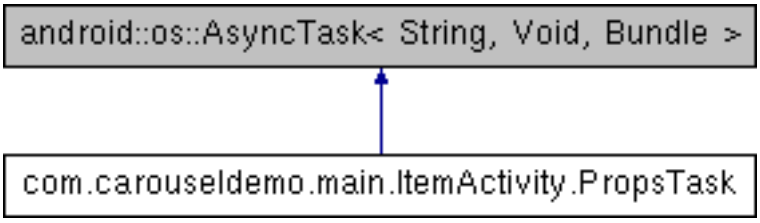
---

**The documentation for this interface was generated from the following file:**

- src/com/carouseldemo/controls/**CarouselAdapter.java**

## com.carouseldemo.main.ItemActivity.PropsTask Class Reference

Inheritance diagram for com.carouseldemo.main.ItemActivity.PropsTask:




### Protected Member Functions

- Bundle **doInBackground** (String...params)
- void **onPreExecute** ()
- void **onProgressUpdate** (Void...unsued)
- void **onPostExecute** (Bundle props)

---

### Member Function Documentation

**Bundle com.carouseldemo.main.ItemActivity.PropsTask.doInBackground (String... *params*)** `[protected]`

**void com.carouseldemo.main.ItemActivity.PropsTask.onPostExecute (Bundle *props*)** `[protected]`

**void com.carouseldemo.main.ItemActivity.PropsTask.onPreExecute ()** `[protected]`

**void com.carouseldemo.main.ItemActivity.PropsTask.onProgressUpdate (Void... *unsued*)** `[protected]`

---

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/main/**ItemActivity.java**

# com.carouseldemo.main.R Class Reference

## Classes

- class **animator**
- class **array**
- class **attr**
- class **color**
- class **drawable**
- class **id**
- class **layout**
- class **string**
- class **styleable**

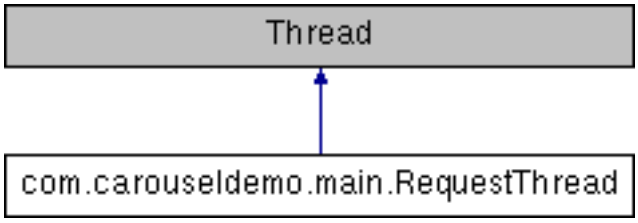
---

The documentation for this class was generated from the following file:

- gen/com/carouseldemo/main/**R.java**

# com.carouseldemo.main.RequestThread Class Reference

Inheritance diagram for com.carouseldemo.main.RequestThread:




## Public Member Functions

- **RequestThread** (Context ctx)
- void **pleaseStop** ()
- void **showDialog** ()
- void **dismissDialog** ()
- void **run** ()
- void **pleasePause** ()
- void **pleaseResume** ()

## Public Attributes

- Integer **flag** = null
- boolean **shouldWork** = true

## Static Public Attributes

- static Handler **handler** = new Handler()

## Private Attributes

- final String **getOfferScript** = "scripts/get_offer.php"
- Object **mPauseLock** = new Object()
- boolean **mPaused** = false
- Dialog **dialog** = null
- Context **context** = null

## Static Private Attributes

- static final String **TAG** = RequestThread.class.getSimpleName()

---

## Constructor & Destructor Documentation

**com.carouseldemo.main.RequestThread.RequestThread (Context *ctx*)**

---

## Member Function Documentation

**void com.carouseldemo.main.RequestThread.dismissDialog ()**

**void com.carouseldemo.main.RequestThread.pleasePause ()**

Call this on pause.

**void com.carouseldemo.main.RequestThread.pleaseResume ()**

Call this on resume.

**void com.carouseldemo.main.RequestThread.pleaseStop ()**
**void com.carouseldemo.main.RequestThread.run ()**
**void com.carouseldemo.main.RequestThread.showDialog ()**

---

## Member Data Documentation

**Context com.carouseldemo.main.RequestThread.context = null** `[private]`
**Dialog com.carouseldemo.main.RequestThread.dialog = null** `[private]`
**Integer com.carouseldemo.main.RequestThread.flag = null**
**final String com.carouseldemo.main.RequestThread.getOfferScript = "scripts/get_offer.php"** `[private]`
**Handler com.carouseldemo.main.RequestThread.handler = new Handler()** `[static]`
**boolean com.carouseldemo.main.RequestThread.mPaused = false** `[private]`
**Object com.carouseldemo.main.RequestThread.mPauseLock = new Object()** `[private]`
**boolean com.carouseldemo.main.RequestThread.shouldWork = true**
**final String com.carouseldemo.main.RequestThread.TAG = RequestThread.class.getSimpleName()** `[static], [private]`

Tag for a class logging

---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/main/**RequestThread.java**

# com.carouseldemo.controls.Rotator Class Reference

## Public Member Functions

- **Rotator** (Context context)
- final boolean **isFinished** ()
- final void **forceFinished** (boolean finished)
- final long **getDuration** ()
- final float **getCurrAngle** ()
- float **getCurrVelocity** ()
- final float **getStartAngle** ()
- int **timePassed** ()
- void **extendDuration** (int extend)
- void **abortAnimation** ()
- boolean **computeAngleOffset** ()
- void **startRotate** (float startAngle, float dAngle, int duration)
- void **startRotate** (float startAngle, float dAngle)
- void **fling** (float velocityAngle)

## Private Attributes

- int **mMode**
- float **mStartAngle**
- float **mCurrAngle**
- long **mStartTime**
- long **mDuration**
- float **mDeltaAngle**
- boolean **mFinished**
- final float **mCoeffVelocity** = 0.05f
- float **mVelocity**
- final float **mDeceleration** = 240.0f

## Static Private Attributes

- static final int **DEFAULT_DURATION** = 250
- static final int **SCROLL_MODE** = 0
- static final int **FLING_MODE** = 1
- static final String **TAG** = Carousel.class.getSimpleName()

---

## Detailed Description

This class encapsulates rotation. The duration of the rotation can be passed in the constructor and specifies the maximum time that the rotation animation should take. Past this time, the rotation is automatically moved to its final stage and computeRotationOffset() will always return false to indicate that scrolling is over.

---

## Constructor & Destructor Documentation

### com.carouseldemo.controls.Rotator.Rotator (Context *context*)

Create a Scroller with the specified interpolator. If the interpolator is null, the default (viscous) interpolator will be used.

---

## Member Function Documentation

**void com.carouseldemo.controls.Rotator.abortAnimation ()**

Stops the animation. Contrary to **forceFinished(boolean)**, aborting the animating cause the scroller to move to the final x and y position

**See Also:**

**forceFinished(boolean)**

**boolean com.carouseldemo.controls.Rotator.computeAngleOffset ()**

Call this when you want to know the new location. If it returns true, the animation is not yet finished. loc will be altered to provide the new location.

**void com.carouseldemo.controls.Rotator.extendDuration (int *extend*)**

Extend the scroll animation. This allows a running animation to scroll further and longer, when used with **setFinalX(int)** or **setFinalY(int)**.

**Parameters:**

| | |
|---|---|
| *extend* | Additional time to scroll in milliseconds. |

**See Also:**

#setFinalX(int)
#setFinalY(int)

**void com.carouseldemo.controls.Rotator.fling (float *velocityAngle*)**

Start scrolling based on a fling gesture. The distance travelled will depend on the initial velocity of the fling.

**Parameters:**

| | |
|---|---|
| *velocityAngle* | Initial velocity of the fling (X) measured in pixels per second. |

**final void com.carouseldemo.controls.Rotator.forceFinished (boolean *finished*)**

Force the finished field to a particular value.

**Parameters:**

| | |
|---|---|
| *finished* | The new finished value. |

**final float com.carouseldemo.controls.Rotator.getCurrAngle ()**

Returns the current X offset in the scroll.

**Returns:**

The new X offset as an absolute distance from the origin.

**float com.carouseldemo.controls.Rotator.getCurrVelocity ()**

Returns the current velocity.

**Returns:**

The original velocity less the deceleration. Result may be negative.

**final long com.carouseldemo.controls.Rotator.getDuration ()**

Returns how long the scroll event will take, in milliseconds.

**Returns:**

The duration of the scroll in milliseconds.

**final float com.carouseldemo.controls.Rotator.getStartAngle ()**

Returns the start X offset in the scroll.

**Returns:**

The start X offset as an absolute distance from the origin.

**final boolean com.carouseldemo.controls.Rotator.isFinished ()**

Returns whether the scroller has finished scrolling.

**Returns:**

True if the scroller has finished scrolling, false otherwise.

**void com.carouseldemo.controls.Rotator.startRotate (float *startAngle*, float *dAngle*, int *duration*)**

Start scrolling by providing a starting point and the distance to travel.

**Parameters:**

| | |
|---|---|
| *startX* | Starting horizontal scroll offset in pixels. Positive numbers will scroll the content to the left. |
| *startY* | Starting vertical scroll offset in pixels. Positive numbers will scroll the content up. |
| *dx* | Horizontal distance to travel. Positive numbers will scroll the content to the left. |
| *dy* | Vertical distance to travel. Positive numbers will scroll the content up. |
| *duration* | Duration of the scroll in milliseconds. |

**void com.carouseldemo.controls.Rotator.startRotate (float *startAngle*, float *dAngle*)**

Start scrolling by providing a starting point and the distance to travel. The scroll will use the default value of 250 milliseconds for the duration.

**Parameters:**

| | |
|---|---|
| *startX* | Starting horizontal scroll offset in pixels. Positive numbers will scroll the content to the left. |
| *startY* | Starting vertical scroll offset in pixels. Positive numbers will scroll the content up. |
| *dx* | Horizontal distance to travel. Positive numbers will scroll the content to the left. |
| *dy* | Vertical distance to travel. Positive numbers will scroll the content up. |

**int com.carouseldemo.controls.Rotator.timePassed ()**

Returns the time elapsed since the beginning of the scrolling.

**Returns:**

The elapsed time in milliseconds.

---

## Member Data Documentation

**final int com.carouseldemo.controls.Rotator.DEFAULT_DURATION = 250** `[static], [private]`
**final int com.carouseldemo.controls.Rotator.FLING_MODE = 1** `[static], [private]`
**final float com.carouseldemo.controls.Rotator.mCoeffVelocity = 0.05f** `[private]`
**float com.carouseldemo.controls.Rotator.mCurrAngle** `[private]`
**final float com.carouseldemo.controls.Rotator.mDeceleration = 240.0f** `[private]`
**float com.carouseldemo.controls.Rotator.mDeltaAngle** `[private]`
**long com.carouseldemo.controls.Rotator.mDuration** `[private]`

**boolean com.carouseldemo.controls.Rotator.mFinished** `[private]`
**int com.carouseldemo.controls.Rotator.mMode** `[private]`
**float com.carouseldemo.controls.Rotator.mStartAngle** `[private]`
**long com.carouseldemo.controls.Rotator.mStartTime** `[private]`
**float com.carouseldemo.controls.Rotator.mVelocity** `[private]`
**final int com.carouseldemo.controls.Rotator.SCROLL_MODE = 0** `[static], [private]`
**final String com.carouseldemo.controls.Rotator.TAG = Carousel.class.getSimpleName()** `[static], [private]`

Tag for a class logging

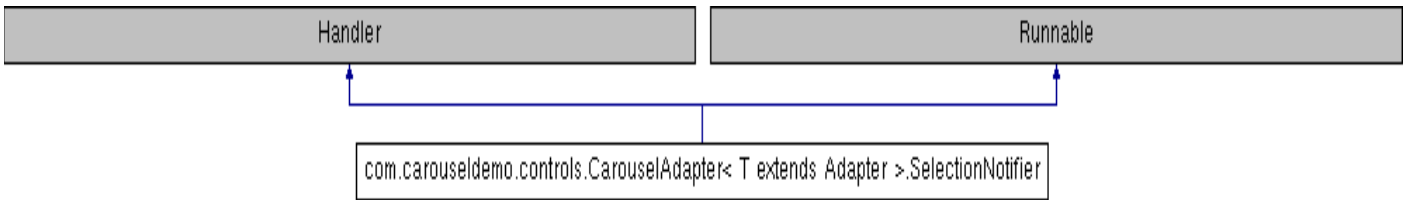
---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/controls/**Rotator.java**

# com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.SelectionNotifier Class Reference

Inheritance diagram for com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.SelectionNotifier:




## Public Member Functions

- void **run** ()

## Member Function Documentation

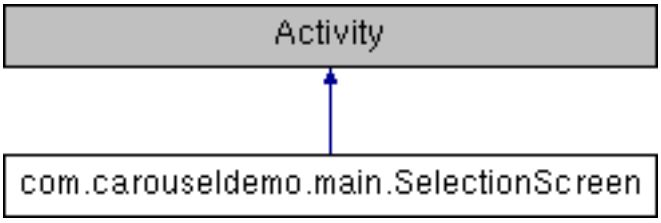
### void com.carouseldemo.controls.CarouselAdapter< T extends Adapter >.SelectionNotifier.run ()

**The documentation for this class was generated from the following file:**

- src/com/carouseldemo/controls/**CarouselAdapter.java**

## com.carouseldemo.main.SelectionScreen Class Reference

Inheritance diagram for com.carouseldemo.main.SelectionScreen:




### Public Member Functions

- void **onDestroy** ()
- void **onBackPressed** ()
- void **onCreate** (Bundle savedInstanceState)

### Static Public Attributes

- static **RequestThread requestThread**

### Protected Member Functions

- void **onActivityResult** (int requestCode, int resultCode, Intent activity)
- void **removeBorder** (ImageView v)
- void **drawBorder** (ImageView v)

### Static Private Attributes

- static final String **TAG** = SelectionScreen.class.getSimpleName()

---

### Member Function Documentation

**void com.carouseldemo.main.SelectionScreen.drawBorder (ImageView *v*)** `[protected]`
**void com.carouseldemo.main.SelectionScreen.onActivityResult (int *requestCode*, int *resultCode*, Intent *activity*)** `[protected]`
**void com.carouseldemo.main.SelectionScreen.onBackPressed ()**
**void com.carouseldemo.main.SelectionScreen.onCreate (Bundle *savedInstanceState*)**
**void com.carouseldemo.main.SelectionScreen.onDestroy ()**
**void com.carouseldemo.main.SelectionScreen.removeBorder (ImageView *v*)** `[protected]`

---

### Member Data Documentation

**RequestThread com.carouseldemo.main.SelectionScreen.requestThread** `[static]`
**final String com.carouseldemo.main.SelectionScreen.TAG = SelectionScreen.class.getSimpleName()** `[static]`, `[private]`

Tag for a class logging

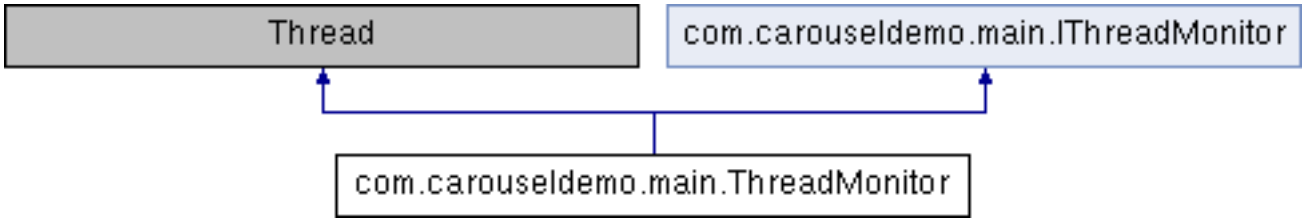
---

**The documentation for this class was generated from the following file:**

·src/com/carouseldemo/main/**SelectionScreen.java**

# com.carouseldemo.main.ThreadMonitor Class Reference

Inheritance diagram for com.carouseldemo.main.ThreadMonitor:




## Public Member Functions

- **ThreadMonitor** (Context ctx, int nCheckPeriodMS)
- synchronized void **setDone** ()
- boolean **setMonitoredReady** ()
- void **setMonitoredThread** (**IMonitorableThread** thr)
- void **run** ()
- synchronized boolean **bark** ()

## Private Member Functions

- void **restartAppActivity** ()
- synchronized void **resetMonitoringLatch** ()

## Private Attributes

- boolean **isDone** = false

## Static Private Attributes

- static CountDownLatch **m_StartMonitoringLatch**
- static CountDownLatch **m_TargetMonitoringLatch**
- static int **m_nTimeoutMS**
- static **IMonitorableThread m_MonitoredThread**

---

## Constructor & Destructor Documentation

**com.carouseldemo.main.ThreadMonitor.ThreadMonitor (Context *ctx*, int *nCheckPeriodMS*)**

---

## Member Function Documentation

**synchronized boolean com.carouseldemo.main.ThreadMonitor.bark ()**

Implements **com.carouseldemo.main.IThreadMonitor** (*p.*).

**synchronized void com.carouseldemo.main.ThreadMonitor.resetMonitoringLatch ()** `[private]`

**void com.carouseldemo.main.ThreadMonitor.restartAppActivity ()** `[private]`

**void com.carouseldemo.main.ThreadMonitor.run ()**

**synchronized void com.carouseldemo.main.ThreadMonitor.setDone ()**

**boolean com.carouseldemo.main.ThreadMonitor.setMonitoredReady ()**

Implements **com.carouseldemo.main.IThreadMonitor** (*p.*).

---

**void com.carouseldemo.main.ThreadMonitor.setMonitoredThread (IMonitorableThread *thr*)**

---

## Member Data Documentation

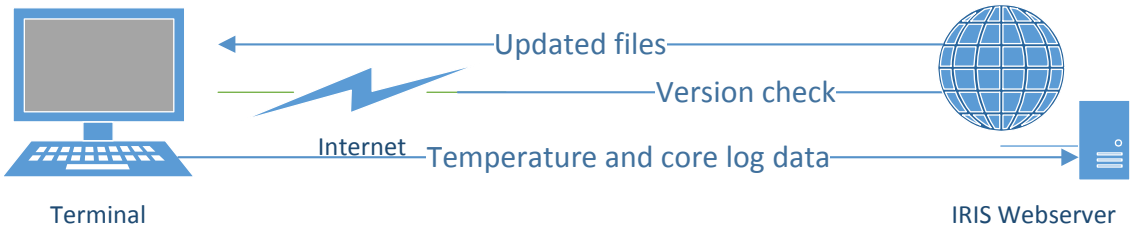
**boolean com.carouseldemo.main.ThreadMonitor.isDone = false** `[private]`
**IMonitorableThread com.carouseldemo.main.ThreadMonitor.m_MonitoredThread** `[static]`, `[private]`
**int com.carouseldemo.main.ThreadMonitor.m_nTimeoutMS** `[static]`, `[private]`
**CountDownLatch com.carouseldemo.main.ThreadMonitor.m_StartMonitoringLatch** `[static]`, `[private]`
**CountDownLatch com.carouseldemo.main.ThreadMonitor.m_TargetMonitoringLatch** `[static]`, `[private]`

---

**The documentation for this class was generated from the following file:**
src/com/carouseldemo/main/**ThreadMonitor.java**

# 6. Appendix B - Controller application

This section provides a description of Wave5CTRL application that is responsible for automatic updates, temperature measurement, system monitoring, logging and brightness control.




Controller application relies on available network connection (either Ethernet or WiFi) to check regularly for updates, download and install them. In case there is no network connection available, then controller will simply log in important data locally.

Along with each update request, controller application also sends important informtaion about temperature levels and usage statistics.

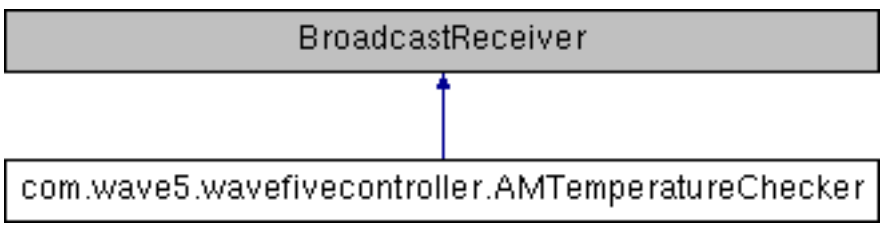
# Technical reference documentation

Technical reference documentation has been generated from Wave5CTRL application sources and contains up to date information of different classes, methods and implemented design patterns.

## Class Documentation

### com.wave5.wavefivecontroller.AMTemperatureChecker Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.AMTemperatureChecker:



## Public Member Functions

- **AMTemperatureChecker** ()
- **AMTemperatureChecker** (**SettingsWrapper** settings)
- void **SetAlarm** (Context context)
- void **CancelAlarm** (Context context)
- void **setInterval** (long interval)
- long **getInterval** ()
- void **onReceive** (Context arg0, Intent arg1)

## Static Public Attributes

- final static String **ONE_TIME** = "onetimeTemp"
- static long **m_nCurrentTemp** = 0

## Private Member Functions

- void **checkEnvTemperatureDriver** ()
- int **getEnvTemperature** ()
- void **setBackLight** (Context ctx, float nDimmingPercent)
- float **getBackLight** (Context ctx)

## Static Private Attributes

- static String **m_TempDriverPath** = "/dev/tcctemp"
- static **SettingsWrapper m_Settings** = null
- static long **m_nInterval** = 1000

---

## Constructor & Destructor Documentation

**com.wave5.wavefivecontroller.AMTemperatureChecker.AMTemperatureChecker ()**
**com.wave5.wavefivecontroller.AMTemperatureChecker.AMTemperatureChecker (SettingsWrapper *settings*)**

---

## Member Function Documentation

**void com.wave5.wavefivecontroller.AMTemperatureChecker.CancelAlarm (Context *context*)**
**void com.wave5.wavefivecontroller.AMTemperatureChecker.checkEnvTemperatureDriver () `[private]`**
**float com.wave5.wavefivecontroller.AMTemperatureChecker.getBackLight (Context *ctx*) `[private]`**
**int com.wave5.wavefivecontroller.AMTemperatureChecker.getEnvTemperature () `[private]`**
**long com.wave5.wavefivecontroller.AMTemperatureChecker.getInterval ()**
**void com.wave5.wavefivecontroller.AMTemperatureChecker.onReceive (Context *arg0*, Intent *arg1*)**
**void com.wave5.wavefivecontroller.AMTemperatureChecker.SetAlarm (Context *context*)**
**void com.wave5.wavefivecontroller.AMTemperatureChecker.setBackLight (Context *ctx*, float *nDimmingPercent*) `[private]`**
**void com.wave5.wavefivecontroller.AMTemperatureChecker.setInterval (long *interval*)**

---

## Member Data Documentation

**long com.wave5.wavefivecontroller.AMTemperatureChecker.m_nCurrentTemp = 0 `[static]`**
**long com.wave5.wavefivecontroller.AMTemperatureChecker.m_nInterval = 1000 `[static], [private]`**
**SettingsWrapper com.wave5.wavefivecontroller.AMTemperatureChecker.m_Settings = null `[static],`**

---

**[private]**
**String com.wave5.wavefivecontroller.AMTemperatureChecker.m_TempDriverPath = "/dev/tcctemp"** **[static], [private]**
**final static String com.wave5.wavefivecontroller.AMTemperatureChecker.ONE_TIME = "onetimeTemp"** **[static]**

---

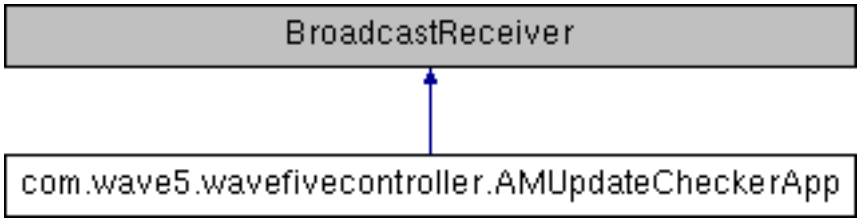
**The documentation for this class was generated from the following file:**

·src/com/wave5/wavefivecontroller/**AMTemperatureChecker.java**

AMTemperatureChecker is used to periodically read “tcctemp” character device (which is actually a temperature kernel driver exposed as characted device). For periodic invokation of temperature checking method, this class derives from Android Alarm.

## com.wave5.wavefivecontroller.AMUpdateCheckerApp Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.AMUpdateCheckerApp:




### Public Member Functions

- **AMUpdateCheckerApp** ()
- **AMUpdateCheckerApp** (**SettingsWrapper** settings)
- void **SetAlarm** (Context context)
- void **CancelAlarm** (Context context)
- void **setInterval** (long interval)
- long **getInterval** ()
- void **onReceive** (Context arg0, Intent arg1)

### Static Public Attributes

- final static String **ONE_TIME** = "onetime"

### Static Private Attributes

- static **SettingsWrapper m_Settings** = null
- static long **m_nInterval** = 600000
- static UpdaterTask **m_UpdaterTask** = null

---

### Constructor & Destructor Documentation

**com.wave5.wavefivecontroller.AMUpdateCheckerApp.AMUpdateCheckerApp ()**
**com.wave5.wavefivecontroller.AMUpdateCheckerApp.AMUpdateCheckerApp (SettingsWrapper *settings*)**

---

### Member Function Documentation

**void com.wave5.wavefivecontroller.AMUpdateCheckerApp.CancelAlarm (Context *context*)**
**long com.wave5.wavefivecontroller.AMUpdateCheckerApp.getInterval ()**
**void com.wave5.wavefivecontroller.AMUpdateCheckerApp.onReceive (Context *arg0*, Intent *arg1*)**
**void com.wave5.wavefivecontroller.AMUpdateCheckerApp.SetAlarm (Context *context*)**
**void com.wave5.wavefivecontroller.AMUpdateCheckerApp.setInterval (long *interval*)**

---

### Member Data Documentation

**long com.wave5.wavefivecontroller.AMUpdateCheckerApp.m_nInterval = 600000** `[static], [private]`
**SettingsWrapper com.wave5.wavefivecontroller.AMUpdateCheckerApp.m_Settings = null** `[static], [private]`
**UpdaterTask com.wave5.wavefivecontroller.AMUpdateCheckerApp.m_UpdaterTask = null** `[static],`

---

`[private]`
**final static String com.wave5.wavefivecontroller.AMUpdateCheckerApp.ONE_TIME = "onetime"** `[static]`

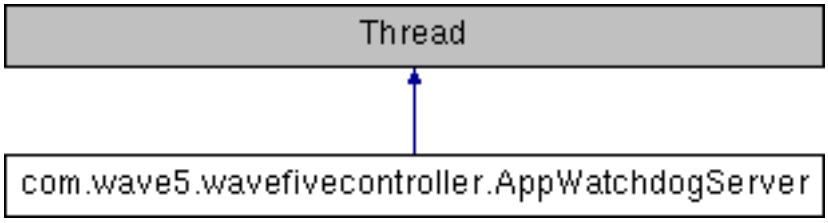
---

**The documentation for this class was generated from the following file:**

- src/com/wave5/wavefivecontroller/**AMUpdateCheckerApp.java**

## com.wave5.wavefivecontroller.AppWatchdogServer Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.AppWatchdogServer:




### Public Member Functions

- **AppWatchdogServer** ()
- void **run** ()
- void **setStopThread** (boolean value)

### Private Attributes

- volatile boolean **stopThread**

### Static Private Attributes

- static String **SOCKET_ADDRESS** = "/wave5ctrl/socket/watchdog"

---

### Constructor & Destructor Documentation

**com.wave5.wavefivecontroller.AppWatchdogServer.AppWatchdogServer ()**

---

### Member Function Documentation

**void com.wave5.wavefivecontroller.AppWatchdogServer.run ()**
**void com.wave5.wavefivecontroller.AppWatchdogServer.setStopThread (boolean *value*)**

---

### Member Data Documentation

**String com.wave5.wavefivecontroller.AppWatchdogServer.SOCKET_ADDRESS = "/wave5ctrl/socket/watchdog"** `[static], [private]`
**volatile boolean com.wave5.wavefivecontroller.AppWatchdogServer.stopThread** `[private]`

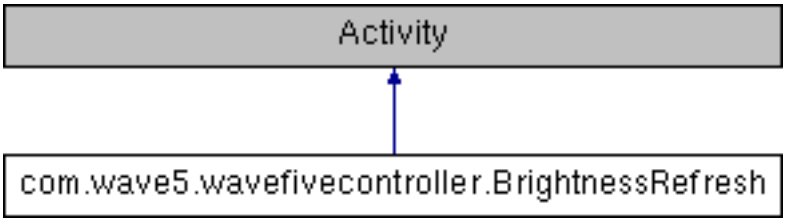
---

**The documentation for this class was generated from the following file:**

- src/com/wave5/wavefivecontroller/**AppWatchdogServer.java**

## com.wave5.wavefivecontroller.BrightnessRefresh Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.BrightnessRefresh:




### Public Member Functions

- void **onCreate** (Bundle savedInstanceState)

---

### Member Function Documentation

**void com.wave5.wavefivecontroller.BrightnessRefresh.onCreate (Bundle *savedInstanceState*)**

---

**The documentation for this class was generated from the following file:**

- src/com/wave5/wavefivecontroller/**BrightnessRefresh.java**

# com.wave5.wavefivecontroller.BuildConfig Class Reference

## Static Public Attributes

·static final boolean **DEBUG** = true

---

## Member Data Documentation

**final boolean com.wave5.wavefivecontroller.BuildConfig.DEBUG = true** `[static]`

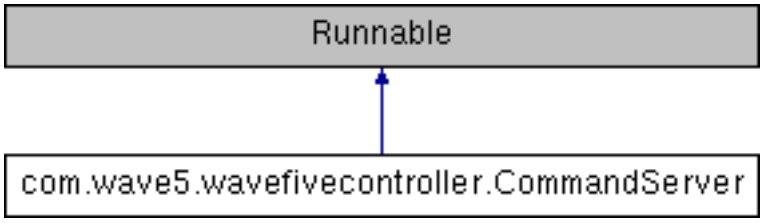
---

**The documentation for this class was generated from the following file:**

·gen/com/wave5/wavefivecontroller/**BuildConfig.java**

# com.wave5.wavefivecontroller.CommandServer Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.CommandServer:




## Public Member Functions

- void **run** ()

## Private Attributes

- String **m_ServerIP** = "127.0.0.1"
- int **m_nPort** = 3333
- Handler **handler** = new Handler()
- ServerSocket **serverSocket**

---

## Member Function Documentation

**void com.wave5.wavefivecontroller.CommandServer.run ()**

---

## Member Data Documentation

**Handler com.wave5.wavefivecontroller.CommandServer.handler = new Handler()** `[private]`
**int com.wave5.wavefivecontroller.CommandServer.m_nPort = 3333** `[private]`
**String com.wave5.wavefivecontroller.CommandServer.m_ServerIP = "127.0.0.1"** `[private]`
**ServerSocket com.wave5.wavefivecontroller.CommandServer.serverSocket** `[private]`

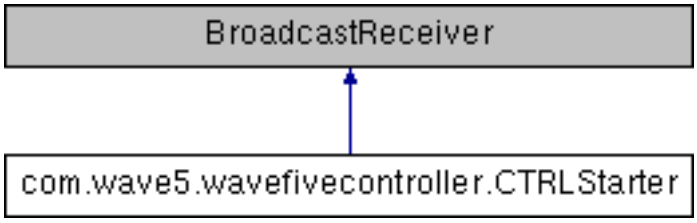
---

**The documentation for this class was generated from the following file:**

- src/com/wave5/wavefivecontroller/**CommandServer.java**

## com.wave5.wavefivecontroller.CTRLStarter Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.CTRLStarter:




### Public Member Functions

- void **onReceive** (Context arg0, Intent arg1)

---

### Member Function Documentation

**void com.wave5.wavefivecontroller.CTRLStarter.onReceive (Context *arg0*, Intent *arg1*)**

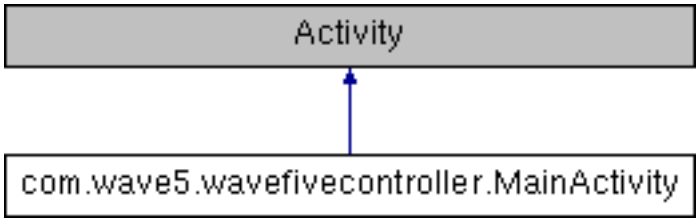
---

**The documentation for this class was generated from the following file:**

- src/com/wave5/wavefivecontroller/**CTRLStarter.java**

## com.wave5.wavefivecontroller.MainActivity Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.MainActivity:




### Public Member Functions

·boolean **onCreateOptionsMenu** (Menu menu)

### Protected Member Functions

·void **onCreate** (Bundle savedInstanceState)

### Private Attributes

·**SettingsWrapper m_SettingsWrapper**
·**AMUpdateCheckerApp m_UpdateCheckerApp**
·**AMTemperatureChecker m_TemperatureChecker**

### Static Private Attributes

·static final int **SCREEN_BRIGHTNESS_MODE_MANUAL** = 0

---

### Member Function Documentation

**void com.wave5.wavefivecontroller.MainActivity.onCreate (Bundle *savedInstanceState*)** `[protected]`
**boolean com.wave5.wavefivecontroller.MainActivity.onCreateOptionsMenu (Menu *menu*)**

---

### Member Data Documentation

**SettingsWrapper com.wave5.wavefivecontroller.MainActivity.m_SettingsWrapper** `[private]`
**AMTemperatureChecker com.wave5.wavefivecontroller.MainActivity.m_TemperatureChecker** `[private]`
**AMUpdateCheckerApp com.wave5.wavefivecontroller.MainActivity.m_UpdateCheckerApp** `[private]`
**final int com.wave5.wavefivecontroller.MainActivity.SCREEN_BRIGHTNESS_MODE_MANUAL = 0** `[static], [private]`

---

**The documentation for this class was generated from the following file:**

·src/com/wave5/wavefivecontroller/**MainActivity.java**

# com.wave5.wavefivecontroller.R Class Reference

## Classes

- class **attr**
- class **drawable**
- class **id**
- class **layout**
- class **menu**
- class **string**
- class **style**

---

The documentation for this class was generated from the following file:

- gen/com/wave5/wavefivecontroller/**R.java**

# com.wave5.wavefivecontroller.SettingsConstants Class Reference

## Static Public Attributes

- static final String **PREFS_NAME** = "WaveFiveController"
- static final String **UPDATE_PERIOD_PREFS** = "UpdateCheckPeriod"
- static final long **UPDATE_PERIOD_DEFAULT** = 300000
- static final String **CAROUSEL_NAME** = "GUIAppName"
- static final String **CAROUSEL_DEFAULT_NAME** = "com.carouseldemo.main"
- static final String **CAROUSEL_VERSION_URL** = "GUIAppVersionURL"
- static final String **CAROUSEL_VERSION_URL_DEFAULT** = "http://78.46.181.171/scripts/gui.php"
- static final String **CAROUSEL_UPDATE_URL** = "GUIAppUpdateURL"
- static final String **CAROUSEL_UPDATE_URL_DEFAULT** = "http://78.46.181.171/scripts/CarouselDemo.apk"
- static final String **CAROUSEL_START_ACTIVITY** = "GUIAppStartActivity"
- static final String **CAROUSEL_START_ACTIVITY_DEFAULT** = "com.carouseldemo.main/.SelectionScreen"
- static final String **TEMPERATURE_TRESHOLD** = "NormalTempLimit"
- static final int **TEMPERATURE_TRESHOLD_DEFAULT** = 70
- static final String **TEMPERATURE_PERIOD_PREFS** = "TemperatureCheckPeriod"
- static final long **TEMPERATURE_PERIOD_DEFAULT** = 120000
- static final String **TEMPERATURE_DIMMING_STEP** = "TemperatureDimmingStepPercent"
- static final float **TEMPERATURE_DIMMING_STEP_DEFAULT** = 0.5f
- static final String **WAVE5CONTROLLER_NAME** = "wavefivecontrollerapp"
- static final String **WAVE5CONTROLLER_NAME_DEFAULT** = "com.wave5.wavefivecontroller"
- static final String **WAVE5CTRL_VERSION_URL** = "W5CTRLAppVersionURL"
- static final String **WAVE5CTRL_VERSION_URL_DEFAULT** = "http://78.46.181.171/scripts/ctrl.php"
- static final String **WAVE5CTRL_UPDATE_URL** = "W5CTRLAppUpdateURL"
- static final String **WAVE5CTRL_UPDATE_URL_DEFAULT** = "http://78.46.181.171/scripts/wavefivecontroller.apk"
- static final String **WAVE5CTRL_START_ACTIVITY** = "W5CTRLStartActivity"
- static final String **WAVE5CTRL_START_ACTIVITY_DEFAULT** = "com.wave5.wavefivecontroller/.**MainActivity**"

---

## Member Data Documentation

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_DEFAULT_NAME = "com.carouseldemo.main"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_NAME = "GUIAppName"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_START_ACTIVITY = "GUIAppStartActivity"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_START_ACTIVITY_DEFAULT = "com.carouseldemo.main/.SelectionScreen"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_UPDATE_URL = "GUIAppUpdateURL"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_UPDATE_URL_DEFAULT = "http://78.46.181.171/scripts/CarouselDemo.apk"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_VERSION_URL = "GUIAppVersionURL"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.CAROUSEL_VERSION_URL_DEFAULT = "http://78.46.181.171/scripts/gui.php"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.PREFS_NAME = "WaveFiveController"** `[static]`

**final String com.wave5.wavefivecontroller.SettingsConstants.TEMPERATURE_DIMMING_STEP = "TemperatureDimmingStepPercent"** `[static]`

**final float com.wave5.wavefivecontroller.SettingsConstants.TEMPERATURE_DIMMING_STEP_DEFAULT = 0.5f** `[static]`

---

**final long com.wave5.wavefivecontroller.SettingsConstants.TEMPERATURE_PERIOD_DEFAULT = 120000** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.TEMPERATURE_PERIOD_PREFS = "TemperatureCheckPeriod"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.TEMPERATURE_TRESHOLD = "NormalTempLimit"** `[static]`
**final int com.wave5.wavefivecontroller.SettingsConstants.TEMPERATURE_TRESHOLD_DEFAULT = 70** `[static]`
**final long com.wave5.wavefivecontroller.SettingsConstants.UPDATE_PERIOD_DEFAULT = 300000** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.UPDATE_PERIOD_PREFS = "UpdateCheckPeriod"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CONTROLLER_NAME = "wavefivecontrollerapp"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CONTROLLER_NAME_DEFAULT = "com.wave5.wavefivecontroller"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CTRL_START_ACTIVITY = "W5CTRLStartActivity"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CTRL_START_ACTIVITY_DEFAULT = "com.wave5.wavefivecontroller/.MainActivity"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CTRL_UPDATE_URL = "W5CTRLAppUpdateURL"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CTRL_UPDATE_URL_DEFAULT = "http://78.46.181.171/scripts/wavefivecontroller.apk"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CTRL_VERSION_URL = "W5CTRLAppVersionURL"** `[static]`
**final String com.wave5.wavefivecontroller.SettingsConstants.WAVE5CTRL_VERSION_URL_DEFAULT = "http://78.46.181.171/scripts/ctrl.php"** `[static]`

---

**The documentation for this class was generated from the following file:**

·src/com/wave5/wavefivecontroller/**SettingsConstants.java**

# com.wave5.wavefivecontroller.SettingsWrapper Class Reference

## Public Member Functions

- **SettingsWrapper** (Context ctx)
- long **getGUIAppUpdatePeriod** ()
- String **getGUIAppUpdateName** ()
- String **getGUIAppVersionURL** ()
- String **getGUIAppUpdateURL** ()
- String **getGUIAppStartActivity** ()
- String **getSelfAppUpdateName** ()
- String **getSelfAppVersionURL** ()
- String **getSelfAppUpdateURL** ()
- String **getSelfAppStartActivity** ()
- long **getTemperatureCheckPeriod** ()
- int **getTemperatureTreshold** ()
- float **getTempDimmingStep** ()
- String **getDeviceId** (Context context)

---

## Constructor & Destructor Documentation

**com.wave5.wavefivecontroller.SettingsWrapper.SettingsWrapper (Context *ctx*)**

---

## Member Function Documentation

**String com.wave5.wavefivecontroller.SettingsWrapper.getDeviceId (Context *context*)**
**String com.wave5.wavefivecontroller.SettingsWrapper.getGUIAppStartActivity ()**
**String com.wave5.wavefivecontroller.SettingsWrapper.getGUIAppUpdateName ()**
**long com.wave5.wavefivecontroller.SettingsWrapper.getGUIAppUpdatePeriod ()**
**String com.wave5.wavefivecontroller.SettingsWrapper.getGUIAppUpdateURL ()**
**String com.wave5.wavefivecontroller.SettingsWrapper.getGUIAppVersionURL ()**
**String com.wave5.wavefivecontroller.SettingsWrapper.getSelfAppStartActivity ()**
**String com.wave5.wavefivecontroller.SettingsWrapper.getSelfAppUpdateName ()**
**String com.wave5.wavefivecontroller.SettingsWrapper.getSelfAppUpdateURL ()**
**String com.wave5.wavefivecontroller.SettingsWrapper.getSelfAppVersionURL ()**
**float com.wave5.wavefivecontroller.SettingsWrapper.getTempDimmingStep ()**
**long com.wave5.wavefivecontroller.SettingsWrapper.getTemperatureCheckPeriod ()**
**int com.wave5.wavefivecontroller.SettingsWrapper.getTemperatureTreshold ()**

---

**The documentation for this class was generated from the following file:**

- src/com/wave5/wavefivecontroller/**SettingsWrapper.java**

# com.wave5.wavefivecontroller.SUDOHelper Class Reference

## Static Public Member Functions

- static boolean **can_execute_root** ()
- static void **execute_as_root** (String[] commands)
- static void **setFileReadable** (String filePath)

---

## Member Function Documentation

**static boolean com.wave5.wavefivecontroller.SUDOHelper.can_execute_root ()** `[static]`

**static void com.wave5.wavefivecontroller.SUDOHelper.execute_as_root (String[]** ***commands*****)** `[static]`

**static void com.wave5.wavefivecontroller.SUDOHelper.setFileReadable (String** ***filePath*****)** `[static]`

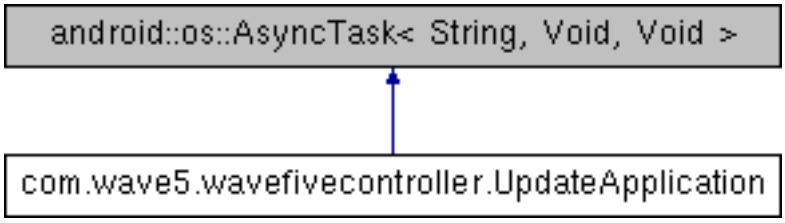
---

**The documentation for this class was generated from the following file:**

- src/com/wave5/wavefivecontroller/**SUDOHelper.java**

# com.wave5.wavefivecontroller.UpdateApplication Class Reference

Inheritance diagram for com.wave5.wavefivecontroller.UpdateApplication:




## Public Member Functions

·void **setContext** (Context contextf)

## Protected Member Functions

·Void **doInBackground** (String...arg0)

## Private Member Functions

·void **execute_as_root** (String[] commands)

## Private Attributes

·Context **context**

---

## Member Function Documentation

**Void com.wave5.wavefivecontroller.UpdateApplication.doInBackground (String... *arg0*) `[protected]`**
**void com.wave5.wavefivecontroller.UpdateApplication.execute_as_root (String[] *commands*) `[private]`**
**void com.wave5.wavefivecontroller.UpdateApplication.setContext (Context *contextf*)**

---

## Member Data Documentation

**Context com.wave5.wavefivecontroller.UpdateApplication.context `[private]`**

---

**The documentation for this class was generated from the following file:**

src/com/wave5/wavefivecontroller/**UpdateApplication.java**