ANT™
the power of less

Creating ANT+ Android Applications

D00001446 Rev 1.3

Copyright Information and Usage Notice

Table of Contents

1 Package contents

AntLib Library

This is an Android library project that must be referenced in order to use the ANT API (See section 3.2 - Referencing the ANTLib_Library project). The API is defined in AntInterface, with AntDefine and AntMesg providing the required constants for ANT communication.

ANT+ Demo source code

This reference application (Eclipse project) shows how to use the ANTLib and configure the ANT radio to connect to a heart rate monitor, a foot pod and a weight scale, and parse the received data. It also shows how to properly claim/release the ANT interface and handle the scenario where the ANT Radio Service has not been installed yet. Paired devices will be remembered until the user selects to pair again, plus proximity search and event buffering can be configured.

.APK's for ANT+ Demo and ANT Radio Service

APK's (Android application packages) are provided so that they can be installed on a phone without Android Market access to test the ANT+ functionality.

2 Other things you will need

All communication with the ANT hardware is achieved through the 'ANT Radio Service'. This application service acts as the proxy between any applications and the system they are running on. It is available from the Android Market for devices with ANT support, and may already be pre-installed on some devices. For other devices ANT functionality may need to be enabled through an update, check with your device manufacture for the most recent update for your device. The ANT Radio Service will only install on a device after the ANT functionality has been enabled.

The ANTLib provides an API for applications to control the ANT chip. This is done through a set of command messages. The ANT host control messaging is documented in the "ANT Message Protocol and Usage" document. This can be found in the Developer's Zone at thisisant.com.

Direct link:
(http://thisisant.com/images/Resources/PDF/1204662412_ant%20message%20protocol%20and%20usage.pdf).

Almost all of the ANT API function calls correspond to a message described in section 9.5 (ANT Message Details). Differences from that document are described in section 4 (ANT Messaging API Functions) below.

The ANT+ Forums provide a place to ask questions regarding ANT development, and give details on common issues you may encounter. The FAQ post on Getting Started with Development of ANT+ Enabled Applications, in the ANT General Questions forum, is essential reading for all developers.

3 Using the Android AntInterface

Outside of the ANT messaging function calls, you will communicate with the ANT Radio Service through:

- Binding with the service
- AntInterface.ServiceListener call backs
- AntInterfaceIntent's

3.1 Check list for ANT support in android applications

This is a general list of requirements for an android application to support ANT.

- Reference the ANTLib_Library. (Section 3.2)
- Add required permissions to the Manifest. (Section 3.4)
- Bind/Unbind the ANT Radio service. (Sections 3.5 and 3.6)
- Claim/Release the interface. (Section 3.10)
- Register Broadcast receivers for status and message intents. (Sections 3.7, 3.8 and 3.9)
- Use the AntInterface object returned by AntInterface.getInstance() to make calls to the ANT API. (Section 4 of this document and Section 9.5 of the "ANT Message Protocol and Usage" document)

3.2 Referencing the ANTLib_Library project

Applications must reference the ANTLib_Library source in order to use the ANT API. For instructions on referencing an android library project using the Eclipse IDE see "Referencing a Library Project" at
http://developer.android.com/guide/developing/projects/projects-eclipse.html

If another build environment is being used the project can be referenced from the command line using the instructions at
http://developer.android.com/guide/developing/projects/projects-cmdline.html

3.3 Checking for ANT support

Applications can check if ANT is supported on the phone before trying to bind to the ANT Radio Service, this is achieved through the call to AntInterface.hasAntSupport().

If your application requires ANT support to be useful and you want to only make it available to devices with ANT support, you can add the following to AndroidManifest.xml:

```
<uses-library android:name="com.dsi.ant.antradio_library" />
```

3.4 Required application permissions

New permissions have been defined in systems with ANT support.

3.4.1 com.dsi.ant.permission.ANT

This permission is required by applications which will be using any ANT Radio Service functionality.

The user will see the following description:

"Allows the application to communicate with ANT devices using the proxy service. Malicious applications may acquire information from, or transmit to, ANT devices without your knowledge."

3.4.2 com.dsi.ant.permission.ANT_ADMIN

This permission is required by any applications which will be using the AntInterface methods to configure the ANT radio.

The user will see the following description:

"Allows the application to directly configure the ANT radio through the proxy service. Malicious applications may prevent other ANT applications from connecting to ANT devices."

3.5 Initializing the ANT Radio Service

The ANT Radio Service will start running when an application attempts to bind with it, which is achieved through the call to AntInterface.initService(). If this returns false, then no binding could be made and the service is not installed.

When you are finished with the ANT Radio Service you should unbind from it, which is achieved with AntInterface.releaseService().

You may repeatedly init and release the service. A new init while the service is connected will result in the existing connection being unbound and a new one bound.

3.5.1 Installing the ANT Radio Service

If it is required, a call to AntInterface.goToMarket() will trigger the Android Market app to open and search for the service installer. If ANT is not supported nothing will be found.

3.6 Tracking service connection state

To know that the service has started up and initialized, you will need to pass an implementation of the AntInterface.ServiceListener interface in to the AntInterface.getInstance() call. The onServiceConnected() and onServiceDisconnected() call backs will then be called to notify any applications of when they can communicate with the service. AntInterface.isServiceConnected() can be used to query this status at any time.

3.7 Tracking the ANT enabled state

Even with a connection to the ANT Radio Service, ANT communication will only occur while ANT is enabled. You must configure a BroadcastReceiver to listen for AntInterfaceIntent.ANT_ENABLED_ACTION and ANT_DISABLED_ACTION. Note that a true result from AntInterface.enable() and disable() will only indicate that the request was sent successfully, and you will have to wait for these Intents for the correct ANT state.

When Airplane mode is entered, the ANT chip will be reset and all subsequent requests will fail. Applications should handle the system intent Intent.ACTION_AIRPLANE_MODE_CHANGED to ensure this is managed smoothly, as even though the ANT_DISABLED_ACTION intent will be sent, it will not be possible to re-enable with a call to enable().

AntInterface.isEnabled() can be used to query this status at any time.

3.8 ANT Resets

Any time a reset of the ANT chip is performed, the AntInerfaceIntent.ANT_RESET_ACTION Intent will be sent. At this point your application should clear any ANT configuration state information, as the ANT Radio has returned to an uninitialized state (as detailed in the ANTResetSystem message documentation), and you will need to reconfigure any channels.

3.9 Receiving ANT messages

All received ANT packets are sent out from the service within an AntInterface.Intent.ANT_RX_MESSAGE_ACTION Intent. The raw data is retrieved with Intent.getByteArrayExtra(AntInterfaceIntent.ANT_MESSAGE), and decoded as documented in "ANT Message Protocol and Usage" (link above) with the values defined in AntMesg and AntDefine.

Also see 3.10.4 Receiving ANT messages with multiple applications.

3.10 Handling multiple applications

Currently any application can bind with the service but only one application may claim control at once. Applications can claim, release, and force claim the ANT Interface. Any ANT messaging requests will fail if another application has control. To support existing applications, if no application has claimed the interface then requests will be accepted from any application.

3.10.1 Claim Interface

AntInterface.claimInterface() will return true if the ANT Interface is available, and control has been given to the current application.

The AntInterfaceIntent.ANT_INTERFACE_CLAIMED_ACTION Intent is sent whenever an application has claimed the ANT Interface. Doing a getIntExtra() for ANT_INTERFACE_CLAIMED_PID provides the process ID of the application which has control.

You can check if you have control at any time by calling hasClaimedInterface().

3.10.2 Release Interface

AntInterface.releaseInterface() gives up control of the ANT Interface, and resets the ANT chip. This will make the ANT Interface available to any other application to claim, or automatically transfer control if another application is in the process of requesting a force claim (see below).

3.10.3 Force Claim Interface

If another application has not given up control of the ANT Interface, you can force ownership to be switched to the current application with AntInterface.requestForceClaimInterface(). The user will always be prompted (in the Notification bar) if they agree to this, and you must call AntInterface.stopRequestForceClaimInterface() when quiting to clear the notification.

If the force claim is allowed, the ANT chip will be reset, and the AntInterfaceIntent.ANT_INTERFACE_CLAIMED_ACTION Intent is sent.

You will not receive any notification if the user chose not to allow the transfer of control.

3.10.4 Receiving ANT messages

While only one application can claim control of the ANT Interface for configuring the chip, ANT_RX_MESSAGE_ACTION will still be broadcast so any application can receive ANT messages. Keep this in mind as it can result in messages being received by your application at any time a BroadcastReceiver is registered to receive the Intent.

4 ANT Messaging API Functions

4.1 Network Selection

4.1.1 ANTAssignChannel

It is not necessary to manually configure a network key (ANTSetNetworkKey) before assigning a channel to a network. Network numbers are mapped to predefined network keys, as specified in Table 1. Network Selection below.

Table 1. Network Selection

Network Number	Network Key
0	Default public
1	ANT+
2	ANT-FS
3+	Invalid: Do not use

4.1.2 ANTSetNetworkKey

If you are a vendor with a private network, please contact Dynastream for information on how this can be configured on Android.

4.2 Event Buffering

4.2.1 ANTConfigEventBuffering

To help preserve battery power, any applications which do not require constant real-time data, can request that events (EVENT_TX, EVENT_RX_FAIL, EVENT_CHANNEL_COLLISION, EVENT_RX (Broadcast Only)) be buffered on the chip for a set time/buffer size before the AntInterface.Intent.ANT_RX_MESSAGE_ACTION intent is triggered for each. The ANTConfigEventBuffering() function allows different values for while the screen is off or on. Note that not all devices will support a timer interval, and a buffer threshold must be set. As any non-buffered events are triggered, the buffer will be flushed.

4.2.2 ANTDisableEventBuffering

ANTDisableEventBuffering() will return the chip to the default setting of all events being forwarded as they arrive.

5 Communicating with ANT+ devices

Once an application is set up to send and receive control messages to the ANT radio, the next step is to configure the ANT radio to communicate with other ANT+ sensors and devices. The data formats and channel parameters for all ANT+ devices are described in ANT+ device profiles. The ANT+ device profiles are available from the ANT+ Adopter Zone on thisisant.com. Registration (no charge) is required to access to the ANT+ Adopter Zone. The ANT+ Simulator, along with its source code, and other useful technical documentation can also be found in the ANT+ Adopter Zone.

6 Use of ANT+ Logos and Certification

The ANT+ logos are used to inform consumers of an application's interoperability with ANT+ devices. Only certified applications are allowed to use the ANT+ name, logos or icons.

Before using any ANT+ branding, the application must complete the ANT+ certification process to ensure that it complies with the device profiles it implements. The process is similar to the certification process for ANT+ sensors and devices but is streamlined for applications. For more details please visit: http://www.thisisant.com/pages/ant/ant-product-certification or contact antalliance@thisisant.com.

Once certification is complete the ANT+ logos can be used on both the application and promotional materials and on the market or other website where the application is available. Logo files will be distributed upon certification.