Android UART Adapter Programming Guide

Version

Version	Author	Date	Digest
0.1	Sean Chang	2015-04-29	Introduce UART Adapter API and working flow.

Contents

Android UART Adapter Programming Guide

Version

Contents

1. The application project 2

2. Application Source Code 3

3. Android NsdManager Class 4

3.1 Class Overview 4

3.2 Class operation flow chart 5

3.3 NsdManager Class API 6

4. UART Adapter working flow 8

4.1 Discover devices 9

4.2 Configure an Ameba device 10

4.3 2-Way char data transmission 11

1. The application project

The UART Adapter(UART Through) application project developed by Android Studio. Android Studio is an integrated development environment (IDE) for developing on the Android platform. An Android developer must install the Android Studio and then they could import and develop this project.

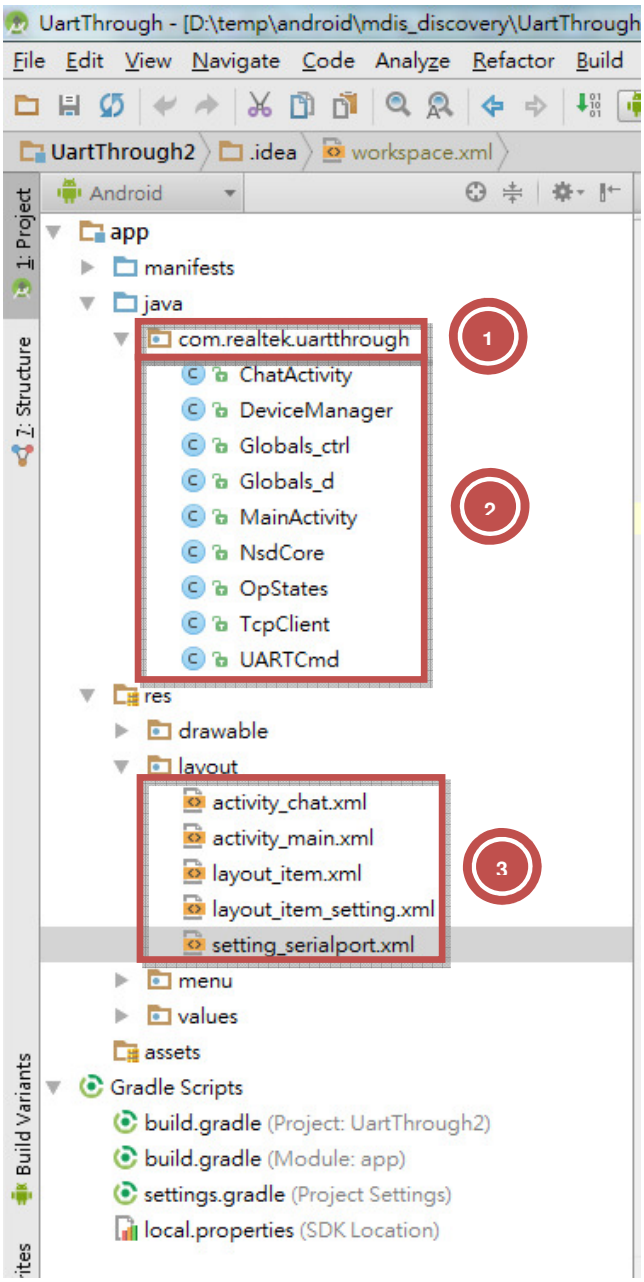


Figure 1-1 Application Structure

Description:

① The main package name;

② *UART Adapter* application source code;

③ Android layout xml file.

2. Application Source Code

The function of each application source code file(② in Figure 1-1) is shown as Table 1-1

Table 1-1 Functions of application source code

File name	Description
ChatActivity.java	The class that implement the updating of chat UI component and status.
DeviceManager.java	The class that implement the management of device core information.
Globals_ctrl.java	The class that implement the core information of control socket.
Globals_d.java	The class that implement the core information of data socket.
MainActivity.java	The main demonstration Activity that calls all the other classes.
NsdCore.java	The class that implement the Network Service Discovery feature.
OpStates.java	The class that implement the status of operation.
TcpClient.java	The class that implement the TCP control and data client.
UARTCmd.java	The class that implement the definition of UART configuration.

3. Android NsdManager Class

java.lang.Object

⇨ android.net.nsd.NsdManager

3.1 Class Overview

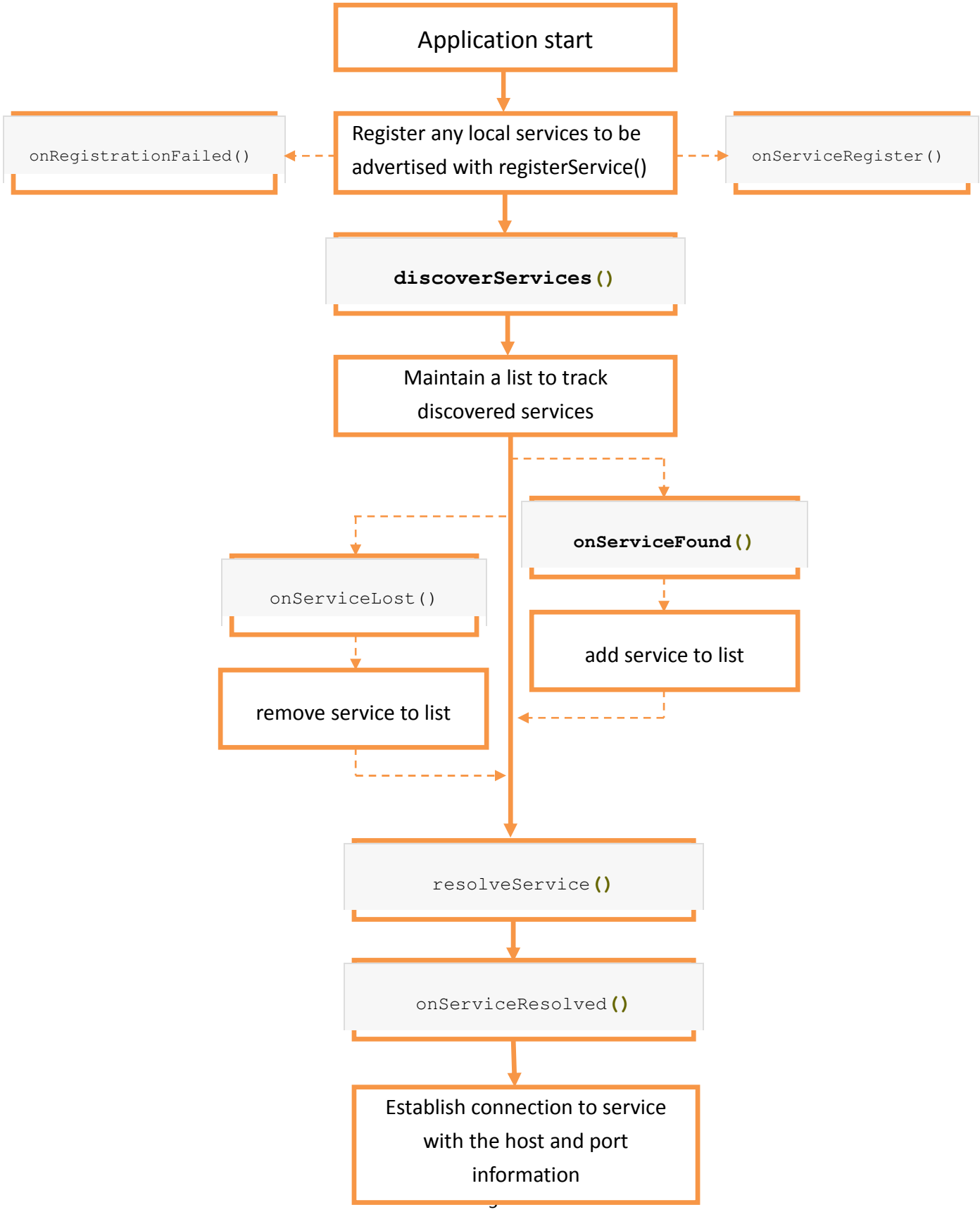
The Network Service Discovery Manager class provides the API to discover services on a network. As an example, if device A and device B are connected over a Wi-Fi network, a game registered on device A can be discovered by a game on device B. Another example use case is an application discovering printers on the network.

The API currently supports DNS based service discovery and discovery is currently limited to a local network over Multicast DNS.

The API is asynchronous and responses to requests from an application are on listener callbacks on a separate thread.

There are three main operations the API supports - registration, discovery and resolution.

3.2 Class operation flow chart



3.3 NsdManager Class API

Nested Classes		
interface	NsdManager.DiscoveryListener	Interface for callback invocation for service discovery
interface	NsdManager.RegistrationListener	Interface for callback invocation for service registration
interface	NsdManager.ResolveListener	Interface for callback invocation for service resolution

Constants		
String	ACTION_NSD_STATE_CHANGED	Broadcast intent action to indicate whether network service discovery is enabled or disabled.
String	EXTRA_NSD_STATE	The lookup key for an int that indicates whether network service discovery is enabled or disabled.
int	FAILURE_ALREADY_ACTIVE	Indicates that the operation failed because it is already active.
int	FAILURE_INTERNAL_ERROR	Failures are passed with onRegistrationFailed(NsdServiceInfo, int), onUnregistrationFailed(NsdServiceInfo, int), onStartDiscoveryFailed(String, int), onStopDiscoveryFailed(String, int) or onResolveFailed(NsdServiceInfo, int).
int	FAILURE_MAX_LIMIT	Indicates that the operation failed because the maximum outstanding requests from the applications have reached.
int	NSD_STATE_DISABLED	Network service discovery is disabled

int	NSD_STATE_ENABLED	Network service discovery is enabled
int	PROTOCOL_DNS_SD	Dns based service discovery protocol

Public Methods	
void	discoverServices(String serviceType, int protocolType, NsdManager.DiscoveryListener listener) Initiate service discovery to browse for instances of a service type.
void	registerService(NsdServiceInfo serviceInfo, int protocolType, NsdManager.RegistrationListener listener) Register a service to be discovered by other services.
void	resolveService(NsdServiceInfo serviceInfo, NsdManager.ResolveListener listener) Resolve a discovered service.
void	stopServiceDiscovery(NsdManager.DiscoveryListener listener) Stop service discovery initiated with `discoverServices(String, int, NsdManager.DiscoveryListener)`.
void	unregisterService(NsdManager.RegistrationListener listener) Unregister a service registered through `registerService(NsdServiceInfo, int, NsdManager.RegistrationListener)`.

4. UART Adapter working flow

UART Adapter tool can be used to:

1. Discover devices;
2. Configure an Ameba device;
3. 2-way char data transmission.

4.1 Discover devices

The working flow of discover devices is shown as Fig 2-1.

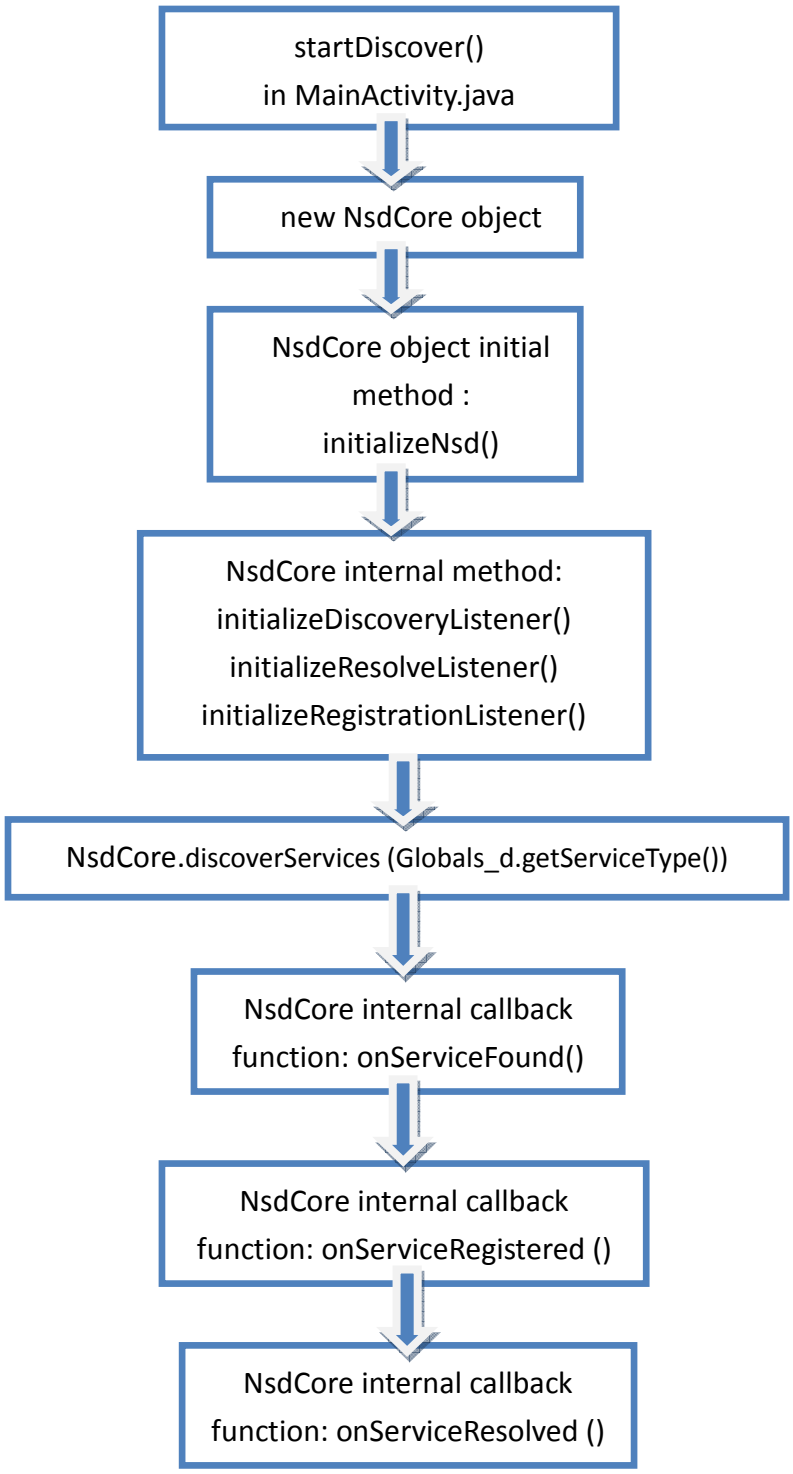


Fig 2-1 The working flow of discover devices

4.2 Configure an Ameba device

The working flow of device configuration is shown as Fig 2-2.

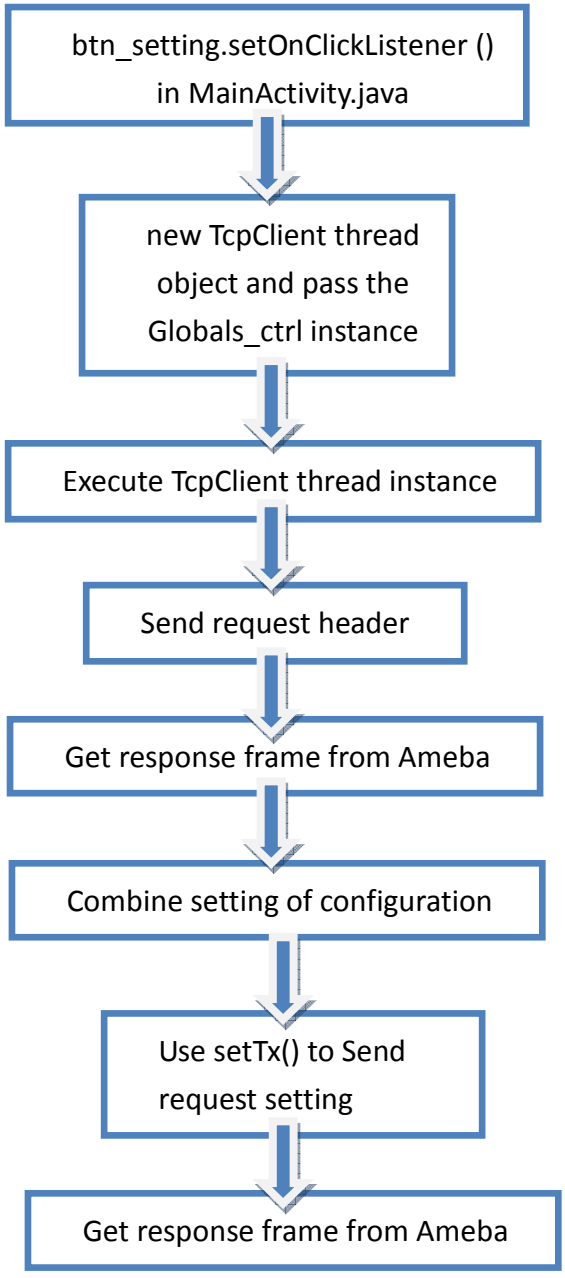


Fig 2-2 The working flow of device configuration

4.3 2-Way char data transmission

The working flow of data transmission is shown as Fig 2-3.

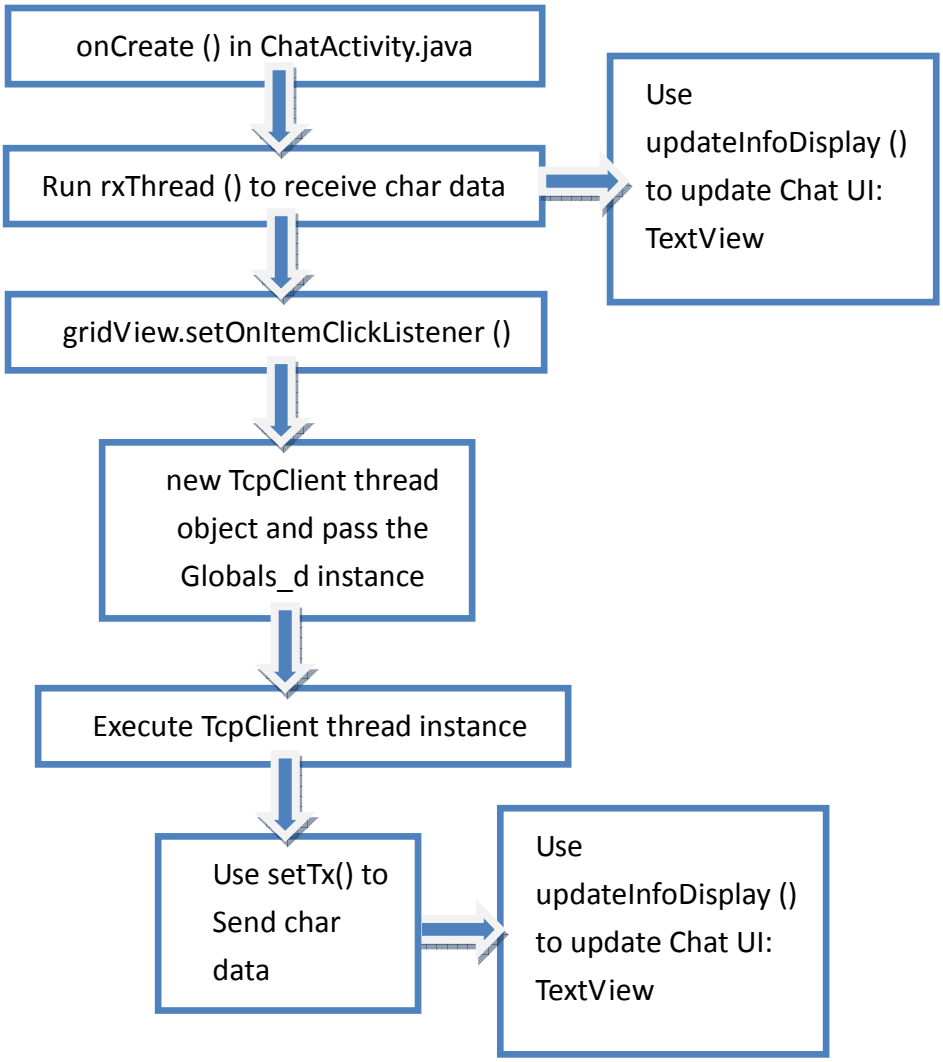


Fig 2-3 The working flow of data transmission.