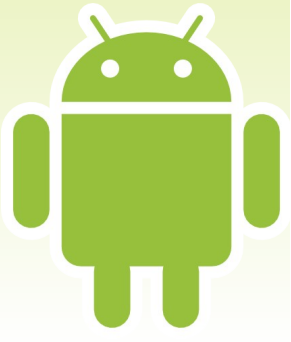
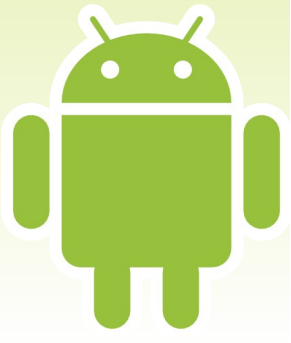
Beaming Data to Devices with NFC



Terminology

- NFC = Near Field Communication
  - Like RFID, but short range (1-4cm)
  - Limited bandwidth
- NDEF = NFC Data Exchange Format
  - ***A*** standard for how data is represented in NFC
  - Came after NFC
    - Result: some NFC items are not NDEF-compliant
    - This course focuses on NDEF



Uses

- Bootstrap
  - Use NFC to coordinate communication on some other channel
    - Bluetooth pairing information
    - WiFi access point configuration
    - Game room to join
  - Takes advantage of no-pairing “low friction” data exchange

Uses

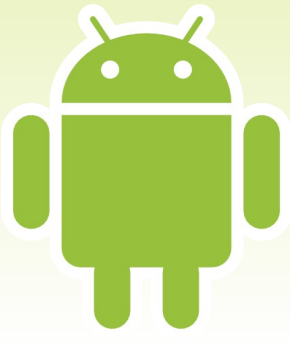
- “Beaming”
  - Really a canned implementation of bootstrap use case
  - Android Beam for pushing files or other payloads from one device to another by tapping their backs
    - As seen in numerous Samsung Galaxy series commercials

Uses

- Miscellany
  - Payment services (tap phone to payment terminal)
  - “Smart posters” as adjunct to QR code

Passive vs. Active

- Passive
  - “Tags” (cards, stickers, key fobs, etc.)
  - Unpowered, static contents
- Active
  - Phones, tablets, payment terminals, etc.
  - Powered, can have an OS behind them

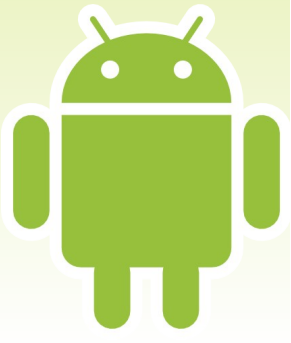


Tag Capabilities

- Capacity Best Measured in Bytes
- Content Management
  - Read-only
  - Write-once
  - Read-write, key protected
  - Read-write, unprotected
- Focus: Read-Write and Write-Once

Tag Compatibility

- NFC Forum Standards
  - Work on all NFC-compliant devices
  - NTAG203 (~137 bytes)
  - Topaz 512 (~450 bytes)
- Mifare Classic
  - Only works with NXP NFC chipsets
    - Not used on Samsung Galaxy S4, most Nexus series
  - Classic 1K (~1024 bytes)



NDEF Structure

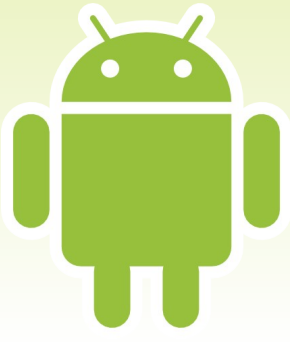
- Message
  - Tag holds one or more of these (typically just one)
- Record
  - Message holds one or more of these (typically just one)
  - Type, sub-type, ID (optional), payload (optional)

NDEF Structure

- Popular Type/Sub-Type Pairs
  - TNF_WELL_KNOWN / RTD_URI for URLs
  - TNF_WELL_KNOWN / RTD_TEXT for plain text
  - TNF_MIME_MEDIA / MIME type
    - Examples: vCard, iCalendar

Reacting to a Tag

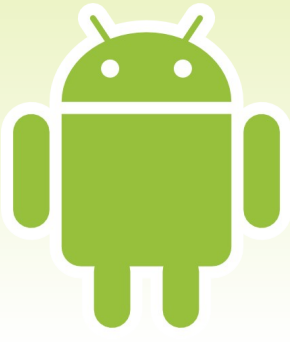
- Step #1: Craft the Intent Filter
  - `ACTION_NDEF_DISCOVERED`
  - Appropriate `<data>` element
    - Typical goal: tight match for your tags
    - Android gives precedence to activities whose `<data>` element more closely match actual tag data



```
<activity
  android:name=".MainActivity"
  android:label="@string/title_activity_main"
  android:launchMode="singleTask">
  <intent-filter>
    <action android:name="android.intent.action.MAIN"/>

    <category android:name="android.intent.category.LAUNCHER"/>
  </intent-filter>
  <intent-filter android:label="@string/app_name">
    <action android:name="android.nfc.action.NDEF_DISCOVERED"/>

    <data android:mimeType="vnd.secret/agent.man"/>

    <category android:name="android.intent.category.DEFAULT"/>
  </intent-filter>
</activity>
```

Reacting to a Tag

- Step #2: Get the Tag Data
  - If Uri, use `getIntent().getData()` as normal
  - If MIME, get the NfcMessage object out of Intent extras
    - Get at the first NfcRecord object
    - Get at the payload from the record
    - Interpret the payload based on the MIME type from `getIntent().getType()`



```
@Override
protected void onNewIntent(Intent i) {
  if (inWriteMode
      && NfcAdapter.ACTION_TAG_DISCOVERED.equals(i.getAction())) {
    writeToTag(i);
  }
  else if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(i.getAction())) {
    readFromTag(i);
  }
}
```

```
void readFromTag(Intent i) {
  Parcelable[] msgs=
      (Parcelable[])i.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);

  if (msgs.length > 0) {
    NdefMessage msg=(NdefMessage)msgs[0];

    if (msg.getRecords().length > 0) {
      NdefRecord rec=msg.getRecords()[0];

      secretMessage.setText(new String(rec.getPayload(), US_ASCII));
    }
  }
}
```

Writing to a Tag

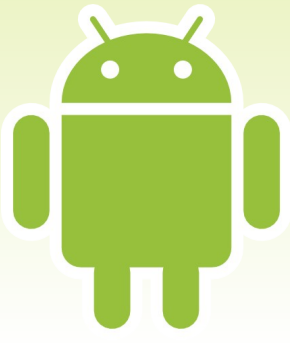
- Step #1: Manifest Adjustements
  - Appropriate `<uses-feature>` element
  - NFC permission
- Step #2: Get an NfcAdapter
  - `NfcAdapter.getDefaultAdapter()`
  - Returns null if device is not NFC-capable

```
<uses-permission android:name="android.permission.NFC"/>

<uses-feature
  android:name="android.hardware.nfc"
  android:required="true"/>
```

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

  nfc=NfcAdapter.getDefaultAdapter(this);
  secretMessage=(EditText)findViewById(R.id.secretMessage);

  nfc.setOnNdefPushCompleteCallback(this, this);

  if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(getIntent().getAction())) {
    readFromTag(getIntent());
  }
}
```

Writing to a Tag

- Step #3: Enable Foreground Dispatch
  - Translation: hand off all tag events to me until otherwise notified, rather than launch some other app
  - Supply IntentFilter objects for the tag events you want, plus a PendingIntent to invoke when those tags are encountered



```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.write_tag:
      setUpWriteMode();
      return(true);

    case R.id.simple_beam:
      enablePush();
      return(true);

    case R.id.file_beam:
      Intent i=new Intent(Intent.ACTION_GET_CONTENT);

      i.setType("*/*");
      startActivityForResult(i, 0);
      return(true);
  }

  return(super.onOptionsItemSelected(item));
}
```

```
void setUpWriteMode() {
  if (!inWriteMode) {
    IntentFilter discovery=
        new IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
    IntentFilter[] tagFilters=new IntentFilter[] { discovery };
    Intent i=
        new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP
            | Intent.FLAG_ACTIVITY_CLEAR_TOP);
    PendingIntent pi=PendingIntent.getActivity(this, 0, i, 0);

    inWriteMode=true;
    nfc.enableForegroundDispatch(this, pi, tagFilters, null);
  }
}
```

Writing to a Tag

- Step #4: Wait for a Tag
  - If same activity, with FLAG_ACTIVITY_CLEAR_TOP and FLAG_ACTIVITY_SINGLE_TOP in PendingIntent, triggers onNewIntent()
- Step #5: Craft the Message
  - NOTE: some NDEF payloads have a particular structure
- Step #6: Write the Message
  - Varies if tag is already NDEF, NDEF-formattable, or unusable

```
void writeToTag(Intent i) {
  Tag tag=i.getParcelableExtra(NfcAdapter.EXTRA_TAG);
  NdefMessage msg=
      new NdefMessage(new NdefRecord[] { buildNdefRecord() });

  new WriteTagTask(this, msg, tag).execute();
}
```

```
class WriteTagTask extends AsyncTask<Void, Void, Void> {
  MainActivity host=null;
  NdefMessage msg=null;
  Tag tag=null;
  String text=null;

  WriteTagTask(MainActivity host, NdefMessage msg, Tag tag) {
    this.host=host;
    this.msg=msg;
    this.tag=tag;
  }
```

```
@Override
protected Void doInBackground(Void... arg0) {
  int size=msg.toByteArray().length;

  try {
    Ndef ndef=Ndef.get(tag);

    if (ndef == null) {
      NdefFormatable formatable=NdefFormatable.get(tag);

      if (formatable != null) {
        try {
          formatable.connect();

          try {
            formatable.format(msg);
          }
          catch (Exception e) {
            text=host.getString(R.string.tag_refused_to_format);
          }
        }
        catch (Exception e) {
          text=host.getString(R.string.tag_refused_to_connect);
        }
        finally {
          formatable.close();
        }
      }
      else {
        text=host.getString(R.string.tag_does_not_support_ndef);
      }
    }
```

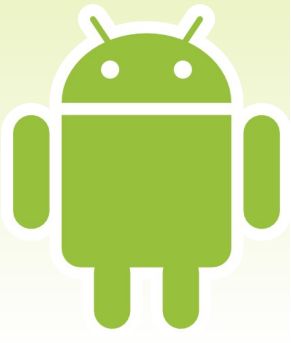
```
    else {
      ndef.connect();

      try {
        if (!ndef.isWritable()) {
          text=host.getString(R.string.tag_is_read_only);
        }
        else if (ndef.getMaxSize() < size) {
          text=host.getString(R.string.message_is_too_big_for_tag);
        }
        else {
          ndef.writeNdefMessage(msg);
          text=host.getString(R.string.success);
        }
      }
      catch (Exception e) {
        text=host.getString(R.string.tag_refused_to_connect);
      }
      finally {
        ndef.close();
      }
    }
  }
  catch (Exception e) {
    Log.e("URLTagger", "Exception when writing tag", e);
    text=host.getString(R.string.general_exception) + e.getMessage();
  }

  return(null);
}
```

```
@Override
protected void onPostExecute(Void unused) {
  host.cleanUpWritingToTag();

  if (text != null) {
    Toast.makeText(host, text, Toast.LENGTH_SHORT).show();
  }
}
```

```
void cleanUpWritingToTag() {
   nfc.disableForegroundDispatch(this);
   inWriteMode=false;
}
```

Android Beam

- NFC P2P
  - Put devices back to back
  - One sends, other receives
- Benefits
  - Devices in hand at all times
  - Capacity



Android Beam

- Implementation
  - Call `setNdefPushMessageCallback()`
  - Called with `createNdefMessage()` at point in time of push
    - You return an NdefMessage

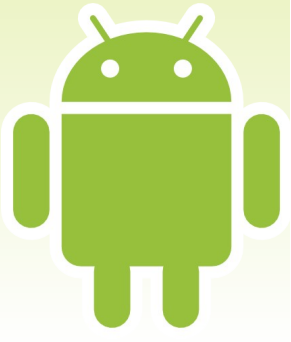
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.write_tag:
      setUpWriteMode();
      return(true);

    case R.id.simple_beam:
      enablePush();
      return(true);

    case R.id.file_beam:
      Intent i=new Intent(Intent.ACTION_GET_CONTENT);

      i.setType("*/*");
      startActivityForResult(i, 0);
      return(true);
  }

  return(super.onOptionsItemSelected(item));
}
```

```
void enablePush() {
  nfc.setNdefPushMessageCallback(this, this);
}
```

```
@Override
public NdefMessage createNdefMessage(NfcEvent event) {
  return(new NdefMessage(
                        new NdefRecord[] {
                            buildNdefRecord(),
                            NdefRecord.createApplicationRecord("com.commonsware.android.jimmyb")
}
```

Android Beam

- AAR: Android Application Record
  - (not to be confused with AAR, the packaged library project from the new Gradle-based build system)
  - Identifies an app that should be able to consume this beam
  - If nothing seems capable of handling the beam, Android will use AAR record to launch Play Store



Android Beam

- NFC Transfer Rate Limited
  - Theoretically 424 kbps
  - In practice, not that fast
- Net: Still Not Good for Large Transfers
  - Capacity gated by time to transfer

Uri Beaming

- Automatic Bluetooth Bootstrapping
  - NFC to pair devices automatically
  - Transfer files, designated by `Uri` values, to the other device
- Benefits
  - Higher transfer rate
  - Longer range

Uri Beaming

- Android 4.1+
- OS Receives the File
  - Not your code
  - No control over where gets written
- Greater Interception Risk
  - Extended range is blessing and curse
  - Encrypt the data if needed

Uri Beaming

- Implementation
  - Option #1: call `setPushBeamUris()` on your NfcAdapter with `file://` or `content://` `Uri[]`
  - Option #2: call `setBeamPushUrisCallback()`
    - Called when beam ready, and you supply `Uri[]` at that point

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.write_tag:
      setUpWriteMode();
      return(true);

    case R.id.simple_beam:
      enablePush();
      return(true);

    case R.id.file_beam:
      Intent i=new Intent(Intent.ACTION_GET_CONTENT);

      i.setType("*/*");
      startActivityForResult(i, 0);
      return(true);
  }

  return(super.onOptionsItemSelected(item));
}
```

```
@TargetApi(16)
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
  if (requestCode==0 && resultCode==RESULT_OK) {
    nfc.setBeamPushUris(new Uri[] {data.getData()}, this);
  }
}
```