# AFD Names & Numbers for Android API

## Integration Guide

July 2022

# Contents

# 1. Introduction

The AFD Names & Numbers for Android API is easy to use and quick to implement in any Android application, while balancing that with providing full flexibility.

An Android Archive file (afdnn-release.aar) containing JAR files and native library files for multiple platforms, provides access to Names & Numbers functionality once added to your project.

# 2. Getting Started

For rapid development and for a quick start, take a look at our sample program to see how Names & Numbers is integrated. When you run that on Android you will need to copy your licence file to your chosen data folder on the device (unless using evaluation data). The data files (names.afd, namesi.afd, etc.) will also need to be present either along with your application or on external storage. If file transfer is more difficult, you can simplify this by transferring the licence file using the SaveLicense method and obtain the latest data files automatically using the update functionality.

To integrate Names & Numbers into your own Project simply add the afdnn-release.aar file to your project libs folder and add a dependency to the Gradle build-file (“`implementation(name:'afdnn-release', ext:'aar')`”). Then create an instance of the afd.nn.Engine class to access AFD Names & Numbers functionality.

On creating an instance of the afd.nn.Engine class you should pass the path to access the product data files from, for example for application private storage:

```
Engine nnEngine = new
afd.nn.Engine(this.getApplicationContext().getExternalFilesDir().toS
tring().concat("/"));
```

Or for an example of an external path:

```
Engine nnEngine = new afd.nn.Engine("/mnt/sdcard/nndata/");
```

For non-evaluation purposes we recommend using the automatic update functionality to obtain the latest data files and using the saveLicense method to save the licence from a Base64 encoded string.

## 3. Using the Sample Application

A sample application demonstrating AFD Names & Numbers functionality is provided with the SDK.

The sample application was developed using Android Studio 4.

To use this load the project into your IDE, build the project and then click Run and follow any prompts to start the application on your device.

The sample uses the “datapath” function in the common.java class to obtain the OS-managed data folder; this is displayed on the “About” screen of the sample application. You may need to change this to point to an appropriate location for your target device.

For Evaluation data, copy that data to a folder on the device, or include it in your apk and ensure you have set the data path correctly as above.

For Full data, either transfer the nn.lic licence file to the device or use the saveLicense method to generate the file from a Base64 string that AFD will have supplied. Then use the checkForUpdates function to obtain the latest data.

# 4. 'How To' Guide

## 4.1 Lookup or Search for Address Records

The addressGetFirst and addressGetNext methods enable address records to quickly be located.

This is typically done as follows:

- Call the clear method to clear any previous search criteria
- Set the appropriate search properties with the criteria to use. (e.g. setSearchLookup to specify a fast-find string to look for, e.g. a postcode)
- Call addressGetFirst to retrieve the first matching record, specifying any required flags (see the function reference).
- If addressGetFirst < 0 report an error, otherwise call addressGetNext repeatedly until AFD_END_OF_SEARCH is returned to return all matching records.
- If the return value is AFD_SUCCESS, read the required properties from the record returned to obtain the required data. If the return value is AFD_RECORD_BREAK no new record data will be returned so, continue to call AddressGetNext

See the appropriate function and property references for full details of these methods and properties.

An example lookup is as follows:

```
// Create an instance of the afd.nn.Engine class
Engine nnEngine = new afd.nn.Engine();

// Set the search criteria
nnEngine.clear();
nnEngine.setSearchLookup("B6 4AA");

// Do the lookup
```

```
int retVal = nnEngine.addressGetFirst(0);

// Deal with any error

if (retVal < 0) {

AlertDialog.Builder dlgAlert = new AlertDialog.Builder(this);
dlgAlert.setTitle("Error");

dlgAlert.setMessage(nnEngine.errorText(retVal));
dlgAlert.create().show();

return;

}

// Retrieve the results

while (retVal >= 0) {

    if (retVal != nnEngine.AFD_RECORD_BREAK) {

            String listItem = nnEngine.getAddressList();

            // do something with this data

    }

    retVal = nnEngine.addressGetNext();

}
```

## 4.2 Retrieve Record

The jumpToRecord method enables records to be quickly retrieved again, (from a list displayed to the user, for example) using its record key.

This is typically done as follows:

- Store the result of the getRecordKey function along with the item at the time of original retrieval.
- Call jumpToRecord with key of the record to be retrieved
- If jumpToRecord < 0 report an error
- If the return value is AFD_SUCCESS, read the required properties from the record returned to obtain the required data.

See the appropriate function and property references for full details of this method and properties.

An example retrieve is as follows, where recordKey is the result of calling getRecordKey for the record originally retrieved:

```
// Create an instance of the afd.nn.Engine class
Engine nnEngine = new afd.nn.Engine();

// Do the lookup
int retVal = nnEngine.jumpToRecord(recordKey);

// Deal with any error
if (retVal < 0) {
      AlertDialog.Builder dlgAlert = new AlertDialog.Builder(this);
      dlgAlert.setTitle("Error");
      dlgAlert.setMessage(nnEngine.errorText(retVal));
      dlgAlert.create().show();
      return;
}

// Retrieve the result
String organisation   = nnEngine.getOrganisation();
String property  = nnEngine.getProperty();
String street         = nnEngine.getStreet();
String locality  = nnEngine.getLocality();
String town           = nnEngine.getTown();
String postcode  = nnEngine.getPostcode();
// do something with this data
```

## 4.3 Saving the License File (Non-Evaluation Only)

To use full data with AFD Names & Numbers, a License will need to be purchased from AFD. This is provided in two forms, a file nn.lic and a Base64 encoded version to make entry or transfer easier.

Either save the nn.lic file in your data folder or use the saveLicense method to pass the supplied base64 encoded text string to apply the license.

The process to do this is typically as follows:

- Obtain the licence data, e.g. by user entry, a file you supply etc. (Do not hard-code this unless you can update your application each year with the new license details).
- Call saveLicense supplying the licence string as a parameter, if this does not return 1 then the string was invalid.
- Test Names & Numbers functionality to ensure the licence is valid.

An example of doing this is as follows:

```
int applied = nnEngine.saveLicense(licenseString);
if (applied == 1) {
            // success
} else {
            // license string not valid
}
```

## 4.4 Updating Data Files (Non-Evaluation Only)

To update Names & Numbers data, the data files (names.afd, namesi.afd, etc.) need to be replaced in your data folder with the latest versions available from our server.

This can be done automatically using instance functions of the afd.nn.Engine class provided for updating.

The process to do this is typically as follows:

- Call checkForUpdate - if this does not return 1 then no update is available so abort
- When convenient call downloadUpdate to commence downloading the file. This function returns a response immediately so that you can continue using the software whilst the download progresses. If it returns
- 0 then no update is available, -1 indicates it couldn't connect to the update server.
- Check periodically by calling isDownloadComplete to determine if the download has completed, when it returns 1 the download has completed, a negative value indicates that an error occurred while downloading .
- Once the download has completed, call applyUpdate at an appropriate time to apply and load the new dataset.

An example of doing this is as follows:

```
int needUpdate = afd.nn.Engine.checkForUpdate();
if (needUpdate == 1) {
      int startedDownload = afd.nn.Engine.downloadUpdate();
      if (startedDownload == 1) {
            while (afd.nn.Engine.isDownloadComplete() == 0) {
               // wait for download to complete
               // can carry on with other tasks
            }
            int updateApplied = afd.nn.Engine.applyUpdate();
      }
}
```

# 5. Function Reference

## 5.1 clear

Clears the search properties ready to start a new search

```
public void clear();
```

This function takes no parameters and returns no value.

## 5.2 addressGetFirst

Starts a lookup or search and returns the first record.

Prior to calling this function, call clear() and set the appropriate search parameters to specify the fields you wish to search for (e.g. setSearchLookup).

```
public int addressGetFirst(int flags)
```

Where flags is an integer and specifies options for the search. These are as follows:

| | |
|---|---|
| AFD_ALL_RECORDS | Return all records |
| AFD_POSTCODE_ONLY | Lookup Postcode Only |
| AFD_PROPERTY_ONLY | Lookup Postcode and Property |

The function returns one of the following values:

| | |
|---|---|
| AFD_SUCCESS | Successful lookup (result returned) |
| AFD_RECORD_BREAK | Search in progress call AddressGetNext to get the next result |
| AFD_INVALID_POSTCODE | Format of postcode supplied is invalid |
| AFD_NOT_FOUND | No matching records found |
| AFD_ERROR_OPENING_FILES | Error opening data files |
| AFD_FILE_READ_ERROR | Error reading data files |
| AFD_DATA_LICENSE_ERROR | Data Licence Error |

If AFD_SUCCESS is returned you can then call any of the properties to obtain the fields for the returned record, e.g., getAddressList.

If AFD_SUCCESS or AFD_RECORD_BREAK is returned, you should call addressGetNext to retrieve each subsequent record until AFD_END_OF_SEARCH is returned.

## 5.3 addressGetNext

Continues a lookup or search, returning the next matching record

```
public int addressGetNext()
```

This function takes no parameters.

The function returns one of the following values:

| | |
|---|---|
| AFD_SUCCESS | Successful lookup (result returned) |
| AFD_RECORD_BREAK | Search in progress call AddressGetNext to get the next result |
| AFD_END_OF_SEARCH | End of Search (no more records to return) |
| AFD_ERROR_OPENING_FILES | Error opening data files |
| AFD_FILE_READ_ERROR | Error reading data files |
| AFD_DATA_LICENSE_ERROR | Data Licence Error |

If AFD_SUCCESS is returned you can then call any of the properties to obtain the fields for the returned record, e.g., getAddressList.

If AFD_SUCCESS or AFD_RECORD_BREAK is returned, you should continue to call AddressGetNext to retrieve each subsequent record until AFD_END_OF_SEARCH is returned.

## 5.4 jumpToRecord

Returns data for the record key supplied

```
public int jumpToRecord(String recordKey)
```

The recordKey parameter is the key of the record to be retrieved. This would have been returned by using the getRecordKey method following a previous call to AddressGetFirst or AddressGetNext. Note this key should not be stored as it is not maintained across data updates; it is designed for re-retrieving a record following a list being provided to the user.

The function returns one of the following values:

| | |
|---|---|
| AFD_SUCCESS | Successful lookup (result returned) |
| AFD_INVALID_RECORD | Key supplied was invalid |
| AFD_ERROR_OPENING_FILES | Error opening data files |
| AFD_FILE_READ_ERROR | Error reading data files |
| AFD_DATA_LICENSE_ERROR | Data Licence Error |

If AFD_SUCCESS is returned, any of the properties can then be called to obtain the fields for the returned record, e.g., getStreet.

## 5.7 errorText

Used to retrieve text to display a friendly error message to go with an error code returned from the search or validation functions (return value less than zero).

```
public String errorText(int errorCode)
```

"errorCode" is an int and should be set to the return value from a previous function call. On return a description for the error is returned which can be stored or displayed for the user.

## 5.9 saveLicense

Used to save a license file on the device from a Base64 encoded string representation.

```
public int saveLicense(string base64License)
```

Base64License is the string containing the base64 encoded license data.

Returns:

| | |
|---|---|
| 0 | Successfully applied license |
| -7 | Data License Error (string is invalid) |

## 5.10 checkForUpdate

Used to check if an update to the Names & Numbers data files is available.

```
public int checkForUpdate()
```

Returns:

| | |
|---|---|
| 1 | Update Available |
| 0 | No Update Available (already on latest data) |
| -1 | Error getting data (normally Internet connection issue) |
| -7 | Data License Error or Evaluation data present |

## 5.11 downloadUpdate

Used to start downloading a data update.

```
public int downloadUpdate()
```

Returns:

| | |
|---|---|
| 1 | Update started downloading |
| 0 | No Update Available (already on latest data) |
| -1 | Error getting data (normally connection issue) |
| -7 | Data License Error or Evaluation data present |

When the function returns data continues to download in the background so your application can continue running. Poll with IsDownloadComplete periodically to determine when the download has finished.

## 5.12 isDownloadComplete

Used to indicate when a data download has completed downloading

```
public int isDownloadComplete()
```

Returns:

| | |
|---|---|
| 1 | Download Complete |
| 0 | Download still in progress |
| -1 | No download started |
| -2 | Download failed (call DownloadUpdate to retry) |

## 5.13 applyUpdate

Used to update data files following a successful download and loads the new dataset.

```
public int applyUpdate()
```

Returns:

| | |
|---|---|
| 1 | Update Successful |
| 0 | Download in progress or failed |
| -1 | No update present to apply |
| -7 | Data License Error or Evaluation data present |

Note that you mustn't call the Names & Numbers functions while this call is completing as old data will be unloaded, the new data unpacked and then loaded during this call. Hence this call does block the thread it is called on which is assumed to be the one you carry out Names & Numbers calls on.

# 6. Properties Reference

Getter and setter methods are provided for the Names & Numbers fields. These are as follows:

| Field Name | Description |
|---|---|
| ***Searchable Only Fields (no get method)*** | |
| SearchLookup | Specify fast-find string to look for, e.g., a postcode or street, town. No other search fields should be set when this is used. |
| Miles | Set distance in miles to search around (for radial searches) |
| Km | Set distance in kilometres to search around |
| ***Result Only Fields (no set method)*** | |
| AddressList | Returns a formatted line for list insertion containing elements of the Address record. Useful when displaying a list of results for the user to choose from. |
| RecordKey | Returns a record key to use for subsequent retrieval of the record with AddressRetrieve. Note this should not be stored as it does not persist across data updates. |
| Name | Returns a formatted Name with title, e.g., "Mr John J Smith" |
| ***Formatted Address Fields*** | |
| Postcode | Returns the Postcode |
| PostcodeFrom | Returns the original postcode if the postcode searched for has been changed (re-coded). |
| Organisation | Returns the Organisation Name |
| Property | Returns the Building Name |

| | |
|---|---|
| Street | Returns the House Number and Street Name |
| Locality | Returns the Locality Name |
| Town | Returns the Post Town for this address |
| ***Raw Address Fields*** *(unformatted with house number in a separate field)* | |
| PAFSubBuilding | The sub building name |
| PAFBuilding | The building name |
| PAFNumber | Property number |
| PAFStreet | The street name |
| PAFLocality | The locality name |
| ***County Fields*** | |
| CountyPostal | Postal County (held, or formally held by Royal Mail) |
| CountyTraditional | Traditional County Name |
| CountyAdministrative | Administrative County Name |
| ***Additional Postal Fields*** | |
| DPS | Postal County (held by Royal Mail) |
| PostcodeType | L for Large User Postcode, S for Small User. |
| MailsortCode | Used for obtaining bulk mail discounts. |
| UDPRN | Royal Mail Unique Delivery Point Reference Number assigned to this letter box. |
| JustBuilt | AFDJustBuilt - Contains the date of inclusion on PAF for properties thought to be recently built. The date is stored numerically in descending format in the form YYYYMMDD. YYYY is the year, MM is the month and DD is the day. For example, 20080304 is 04/03/2008. |
| | |

| | |
|---|---|
| ***Name Fields*** | |
| Gender | The gender (M or F) of the resident if known. |
| Forename | The first name of the resident |
| MiddleInitial | The middle initiate of the resident. |
| Surname | The surname/last name of the resident. |
| OnEditedRoll | Indicates if the resident is on the edited electoral roll (i.e., they have not opted out). Set to Y if they are on the Edited Roll, N if not, blank for Organisation and other records). To search set to #Y to return only records on the electoral roll, #N only for those not on the electoral roll or !N for all records including Organisations but excluding those not on the Edited Roll. |
| DateOfBirth | The residents date of birth if known (electoral roll attainers in the last 10 years only). |
| Residency | Gives time in years that the occupant has lived at this address. |
| HouseholdComposition | Describes the household composition of the selected address [4] |
| ***Additional Organisation Information Fields*** | |
| Business | Provides a description of the type of business |
| SICCode | Standard Industry Classification Code for an organisation record. |
| Size | Gives an indication of the number of employees of an organisation. [5] |

| | |
|---|---|
| ***Phone Number Related Fields*** | |
| Phone | Phone Number |
| ***Geographical Fields*** | |
| GridE | Grid Easting as a 6-digit reference |
| GridN | Grid Northing as a 6/7-digit reference |
| Latitude | Latitude representation of Grid Reference in Decimal Format (WGS84) |
| Longitude | Longitude representation of Grid Reference in Decimal Format (WGS84) |
| UrbanRuralCode | Provides a code which indicates if an area is mainly urban or rural [3] and how sparsely populated those areas are. |
| UrbanRuralName | Provides a description which goes along with the UrbanRuralCode. [3] |
| ***Administrative Fields*** | |
| WardCode | Code identifying the electoral ward for this postcode |
| WardName | Name identifying the electoral ward for this postcode |
| AuthorityCode | Local/Unitary Authority for this Postcode (same as the start of the ward code). |
| AuthorityName | Local / Unitary Authority for this postcode |
| ConstituencyCode | Code for the Parliamentary Constituency for this postcode |
| ConstituencyName | Name of the Parliamentary Constituency for this postcode |
| EERCode | Code identifying the European Electoral Region for this postcode |

| | |
|---|---|
| EERName | Name identifying the European Electoral Region for this postcode |
| LEACode | Code identifying the Local Education Authority for this postcode |
| LEAName | Name identifying the Local Education Authority for this postcode |
| TVRegion | ISBA TV Region (not TV Company) |
| ***Postcode Level Property Indicator Fields*** | |
| Occupancy | Indication of the type of occupants of properties found on the selected postcode [1] |
| AddressType | Indication of the type of property level data to capture to have the full address for a property on the selected postcode. [2] |
| ***NHS Fields*** | |
| NHSCode | National Health Service Area Code |
| NHSName | National Health Service Area Name |
| NHSRegionCode | National Health Service Region Code |
| NHSRegionName | National Health Service Region Name |
| PCTCode | National Health Service Primary Care Trust Code for England (Local Health Board Code in Wales, Community Health Partnership in Scotland) |
| PCTName | Name matching the PCT Code field |
| ***Censation Fields*** | |
| CensationCode | Censation Code assigned to this Postcode |
| Affluence | Affluence description |
| LifeStage | LifeStage description |

| | |
|---|---|
| AdditionalCensusInfo | Additional information derived from the Census |

**Notes**:

[1] Possible Occupancy values and descriptions are as follows (information in brackets not part of the description):

1. Large User Organisation (Single Organisation on this postcode)
2. Small User Organisation (All the properties on this postcode are likely to be businesses)
3. Mostly Organisations (Most of the properties on this postcode are organisations)
4. Mixed (This postcode contains a mixture of business and residential addresses)
5. Mostly Residential (Most of the properties on this postcode are residential)
6. Residential (All the properties on this postcode are likely to be residential)

[2] Possible Address Type values and descriptions are as follows (information in brackets not part of the description):

1. Numbered (Only a property number needs to be captured)
1. 2. Numbered and Named (This postcode contains a mixture of properties needing a property number and those needing a property name including properties, such as 16b)
2. 3. Numbered and Named, Likelihood of Multiple Occupancy (This postcode contains a mixture of properties needing a property number and those needing a property name. Some of the properties on this postcode are likely to contain multiple occupants, e.g., flats).
3. Named (This postcode only contains properties needing a property name).

4. Non-Standard Address Format (This refers to addresses which do not have a street field at all or have multiple street names on the same postcode. This also includes addresses with numbered localities (no street but a house number which goes in with the locality field). It is in-effect a warning to be careful in capturing the property information as it is not in one of the most common address formats).
5. PO Box (This postcode has a PO Box number)
6. 7. No Property Information (Addresses on this postcode have no property information - i.e., capture an Organisation or Resident name only)

[3] The Urban Rural Code differs from England and Wales, Scotland, and Northern Ireland. The possible codes and their meanings are as follows:

**England & Wales**

1. Urban (Sparse): Falls within Urban settlements with a population of 10,000 or more and the wider surrounding area is sparsely populated
2. Town and Fringe (Sparse): Falls within the Small Town and Fringe areas category and the wider surrounding area is sparsely populated.
3. Village (Sparse): Falls within the Village category and the wider surrounding area is sparsely populated.
4. Hamlet and Isolated Dwelling (Sparse): Falls within the Hamlet and Isolated Dwelling category and the wider surrounding area is sparsely populated.
5. Urban (Less Sparse): Falls within urban settlements with a population of 10,000 or more and the wider surrounding area is less sparsely populated.
6. Town and Fringe (Less Sparse): Falls within the Small Town and Fringe areas category and the wider surrounding area is less sparsely populated.

7. Village (Less Sparse): Falls within the village category and the wider surrounding area is less sparsely populated.
8. Hamlet and Isolated Dwelling (Less Sparse): Falls within the Hamlet & Isolated Dwelling category and the wider surrounding area is less sparsely populated

**Scotland**

S1. Large Urban Area: Settlement of over 125,000 people.

S2. Other Urban Area: Settlement of 10,000 to 125,000 people.

S3. Accessible Small Town: Settlement of 3,000 to 10,000 people, within 30 minutes' drive of a settlement of 10,000 or more.

S4. Remote Small Town: Settlement of 3,000 to 10,000 people, with a drive time of 30 to 60 minutes to a settlement of 10,000 or more.

S5. Very Remote Small Town: Settlement of 3,000 to 10,000 people, with a drive time of over 60 minutes to a settlement of 10,000 or more.

S6. Accessible Rural: Settlement of less than 3,000 people, within 30 minutes' drive of a settlement of 10,000 or more.

S7. Remote Rural: Settlement of less than 3,000 people, with a drive time of 30 to 60 minutes to a settlement of 10,000 or more.

S8. Very Remote Rural: Settlement of less than 3,000 people, with a drive time of over 60 minutes to a settlement of 10,000 or more.

**Northern Ireland**

A - E (Urban):

A. Belfast Metropolitan Urban Area

B. Derry Urban Area

C. Large Town: 18,000 and under 75,000 people

D. Medium Town: 10,000 and under 18,000 people

E. Small Town: 4,500 and under 10,000 people

F - H (Rural):

F. Intermediate Settlement: 2,250 and under 4,500 people

G. Village: 1,000 and under 2,250 people

H. Small Village, Hamlet, or Open Countryside: Less than 1,000

[4] The household composition field includes both a number and description and can have any of the following values.

1. 1 Male and 1 Female occupant with different surnames
2. 1 Male and 1 Female occupant with the same surname (married couples)
3. Mixed household
4. More than 2 persons with the same surname (e.g., older families).
5. 1 Male Occupant Only
6. 1 Female Occupant Only
7. More than 7 persons (e.g., old people's home).

[5] The Size property can have any of the following values:

A. 1 to 9 employees

B. 10 to 19 employees

C. 20 to 49 employees

D. 50 to 99 employees

E. 100 to 199 employees

F. 200 to 499 employees

G. 500 to 999 employees

H. 1000+

(If blank, then this is unknown or not applicable).