Digits NFC Toolkit 1.6.3

This plugin provides the tools for reading and writing to and from NFC (NDEF) tags. Using the global API the native NFC functionalities of Android & iOS can be accessed. All Record types in the NDEF standard (Text, Uri, Smart Poster, Mime Media,...) are supported. More about all the formats of NDEF in the following pages.

Platforms:

	Android	iOS
Read NFC Tag Info	Yes	Yes (requires iOS 13 or newer), only in foreground
Read NFC Content (NDEF Message)	Yes Calling “Enable” in API will keep checking for NFC tags	Yes, only in foreground. Pop-up will appear when calling “Enable” in the API and need to call “Enable” again to read again
Write NFC Content (NDEF Message)	Yes	Yes (requires iOS 13 or newer), only in foreground. Will need to call “RequestNDEFWrite” first and a write pop-up will appear when calling “Enable” afterwards in the API
Push NFC Content (NDEF Message)	Yes, uses Host Card Emulation on Android 10+ and Android Beam pre Android 10	No, but can be receiver of NDEF Message pushed from Android device
Make Tag Readonly	Yes	No

Table of contents

General

NDEF Format

NDEF(NFC Data Exchange Format) is the transfer protocol for most NFC tags. NFC tags can holds multiple data records (NDEF Records). A collection of these records is stored in the container (NDEF Message). The NDEF Message is what gets read/written when holding a NFC tag against your NFC device.

NFC Tag Info

The info of the tag itself, such as the tag id.

NDEF Message

This acts as a container for NDEF records.

NDEF Record

This contains a particular type of content.
Following types exist in the NDEF Standard:

Well Known Record Types:
- Text
- Uri
- SmartPoster

Other Record Types:
- Absolute Uri
- Empty
- External Type
- Mime Media
- Unknown

More about each type of record in the following pages.

Examples

See the Samples directory of the plugin

Basic usage

General

All the API methods call be accessed from the NativeNFCManager class and can be called from any class. For example: *NativeNFCManager.IsNDEFWriteSupported()*

iOS Note: You need to configure a provision profile with "NFC Tag Reading" enabled to allow your app to access the NFC functionality of your device.
(See https://www.appcoda.com/corenfc-introduction/ Section: *Setting up App Entitlements and Privacy*)

If you want to check if NFC is currently enabled on the device, use the IsNFCEnabled() method, so you can notify the user to enable it in the device settings when needed.

Read operations

To start reading information from your NFC Tag make sure the device supports NFC Read operations. The following API methods can be used to check which Read operations are supported:

- *IsNFCTagInfoReadSupported:* Checks if NFC Tag Info Read is supported on this device (for example: the id of the tag itself)
- *IsNDEFReadSupported:* Checks if NDEF Read is supported on this device (for example: a data record with some text)

To show the NFC settings of the device (Android only) you can use the API method: *ShowNFCSettings*

To start a Read Operation call Enable(). The will tell the device to start checking for NFC tags. The behaviour is slightly different on the different platforms:

- Android: The device will keep checking for NFC tags (in the background)
- iOS: The device will check for a NFC tag once (in the foreground). A native pop-up will appear when calling this method. This behaviour is defined by iOS itself. After the read operation is finished you need to call Enable() again to read again.
  The read request will timeout if you wait too long to scan the NFC tag. If you want to continue reading when this happens, set the ResetOnTimeout property to true.
  NOTE: Some NFC tags with older formats require the old reader session of iOS. To enable this set the UseNDEFReaderSession property to true. This reader session doesn't support reading the tag info, only the NDEF contents.

To get the data read by the device you need to register event listeners before starting the read operation. The following API methods can be used to register an event listener for read operations:

- *AddNFCTagDetectedListener*: Adds given OnNFCTagDetected listener.
  The format for the listener is: OnNFCTagDetected(NFCTag tag)
  The tag parameter will contains the info about the tag itself.
  NOTE: This event will not be executed if *IsNFCTagInfoReadSupported()* returns false.
- *AddNDEFReadFinishedListener*: Adds given OnNDEFReadFinished listener
  The format for the listener is: OnNDEFReadFinished(NDEFReadResult result)
  The result parameter contains information about whether the read operation was a success and has the NDEFMessage that been read from the tag.
  NOTE: this event will not be executed if *IsNDEFReadSupported()* returns false.

Write operations

To start writing data to your NFC Tag make sure the device supports NFC Write operations. The following API methods can be used to check which Write operations are supported:

- *IsNDEFWriteSupported:* Checks if NDEF Write is supported on this device
  (for example: some data records containing text)

To start a Write Operation call *RequestNDEFWrite()* and pass it the NDEFMessage you want to write as a parameter. The NDEFMessage will be put in the pending state until a NFC Tag gets detected.
(Android: Make sure to call Enable() if it wasn't enabled already or disabled using Disable()).
(iOS: Call Enable() after this method to show a pop-up to write pending NDEFMessage)
When a NFC Tag gets detected it will try to write that pending NDEFMessage.

To get the results of the write operation you need to register event listeners before starting the write operation. The following API methods can be used to register an event listener for write operations:

- *AddNDEFWriteFinishedListener*: Adds given OnNDEFWriteFinished listener
  The format for the listener is: OnNDEFWriteFinished(NDEFWriteResult result)
  The result parameter contains information about whether the write operation was a success and has the NDEFMessage that been written to the tag.
  NOTE: this event will not be executed if *IsNDEFWriteSupported ()* returns false.

Push operations

To send data between two Android devices (or from Android device to iOS device) use the "Push" methods of the plugin. They are similar to the "Write" methods.
NOTE: Google has deprecated Android Beam on Android 10 or newer. The new system uses NFC Host Card Emulation to send NFC content to the other device, so the sending device should support the FEATURE_NFC_HOST_CARD_EMULATION feature.

- *IsNDEFPushSupported*: Checks if NDEF Push is supported on this device
- *IsNDEFPushEnabled*: Checks if NDEF Push is enabled on this device
- *RequestNDEFPush*: Will try to push given NDEF Message when another device is being held against this device
- *AddNDEFPushFinishedListener*: Adds given OnNDEFPushFinished listener
  The format for the listener is: OnNDEFPushFinished(NDEFPushResult result)

Other operations

- *RequestNDEFMakeReadonly* (Android only): Starts a make readonly request. After calling this method you can hold a tag against the device and it will try to make it readonly.
- *AddNDEFMakeReadonlyFinishedListener*: Adds given OnNDEFMakeReadonlyFinished listener
  The format for the listener is: OnNDEFMakeReadonlyFinished(NDEFMakeReadonlyResult result)
  The result parameter contains information about whether the make readonly operation was a success.

NFC Tag Info

Description

This information of the tag itself

Properties

- (String) id: The id of the tag
- (NFCTechnology[]) technologies: List of technologies supported by the tag
- (String) manufacturer: Name of the manufacturer
- (Boolean) writable: Indicates if this tag is writable
- (int) maxWriteSize: The max write size (in bytes) if writable

Example

- Id: 4ED35AF482B1
- Technologies: NFCA, MIFARE_ULTRALIGHT, NDEF
- Writable: true
- MaxWriteSize: 137

Well Known Record Types

Text Record

Description

This record contains text

Properties

- (String) text: The text value
- (String) languageCode: The (2 character) language code
- (TextEncoding) textEncoding: The text encoding

Example

- text: Hello world
- languageCode: en
- textEncoding: UTF8

Uri Record

Description

This record contains an uri

Properties

- (String) fullUri: The full uri
- (String) uri: The uri part of the full uri (plugin determines this automatically)
- (String) protocol: The abbreviated part of the full uri (plugin determines this automatically)

Example

- fullUri: http://www.test.com
- uri: test.com
- protocol: http://www (This is 1 byte in the record)

Smart Poster Record

Description

This record is essentially an Uri Record with more metadata

Properties

- (UriRecord) uriRecord: The main uri record
- (TextRecord[]) titleRecords: The title records (1 per language code allowed)
- (MimeMediaRecord[]) iconRecords: The icon records
- (NDEFRecord[]) extraRecords: The other records
- (RecommendedAction) action: Specifies the recommended action that should be taken
- (int)size: The size of the file referenced by the uri
- (string)mimeType: The mime type of the file referenced by the uri

Example

- uriRecord: <main uri record pointing to a png file>
- titleRecord: <1 or move TextRecords with different language code>
- iconRecord: <MimeMediaRecord for icon file>
- extraRecord: <Extra records>
- action: OPEN_FOR_EDITING
- size: 1487
- mimeType: image/png

Other Record Types

Absolute Uri Record

Description

This record contains an absolute uri.

Properties

- (String) uri: The absolute uri

Example

- Uri: http://google.com

Empty Record

Description

This record is empty and does not hold any content.

Properties

- None

Example

- None

External Type Record

Description

This record refers to an external type not defined by the NDEF standard. This can be for example a value being send to a particular app using the bundle identifier.

Properties

- (string) DomainName: The name of the domain
- (string) DomainType: The type of the domain
- (byte[]) DomainData: The data that needs be send to the domain

Example

- DomainName: com.company.testapp
- DomainType: MainClass
- DomainData: 4ABCLP319S (as bytes)

Mime Media Record

Description

This record refers to a media file

Properties

- (string) MimeType: The type of the media file in mime format
- (byte[]) MimeData: The data of the media file

Example

- MimeType: image/png
- MimeData: <bytes of png file>

Unknown Record

Description

This record defines unknown data

Properties

- None

Example:

- None