

[HOME](#) > [ANDROID](#) > [BASIC](#)

# 안드로이드 - DownloadManager로 파일 다운받는 방법

JS

[Follow](#)

13 Jan 2019

DownloadManager는 HTTP 파일을 다운받는데 도와주는 시스템 서비스입니다. 앱은 저장될 위치의 URI와 다운받을 HTTP URI를 DownloadManager에 알려주기만 하면 알아서 받아줍니다. 또한, DownloadManager 내부의 백그라운드 서비스에서 다운을 받기 때문에 앱에서 쓰레드를 생성할 필요는 없습니다.

다운로드매니저는 noti피케이션에 다운로드 상태를 보여주고, 완료가 되면 브로드캐스트로 완료되었음을 알려줍니다. 또한 실시간으로 다운로드 상태를 체크할 수도 있습니다.

장점을 정리하면 다음과 같습니다.

- 앱에서 다운로드를 위한 백그라운드 쓰레드를 만들 필요가 없습니다.
- noti피케이션을 따로 구성할 필요가 없습니다.
- 다운로드가 완료되면 브로드캐스트로 알려줍니다.
- 불안정한 네트워크 상태에 대한 예외처리가 되어있습니다. 다운로드를 실패하는 경우 다시 시도할 수 있습니다.

어떻게 다운로드매니저를 사용하는지 예제를 통해 알아보겠습니다.

완성된 예제는 [GitHub](#)에 있습니다.

## UI 구성

다운로드 요청, 상태 확인, 취소 처리를 하기 위해 버튼 3개를 만들었습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

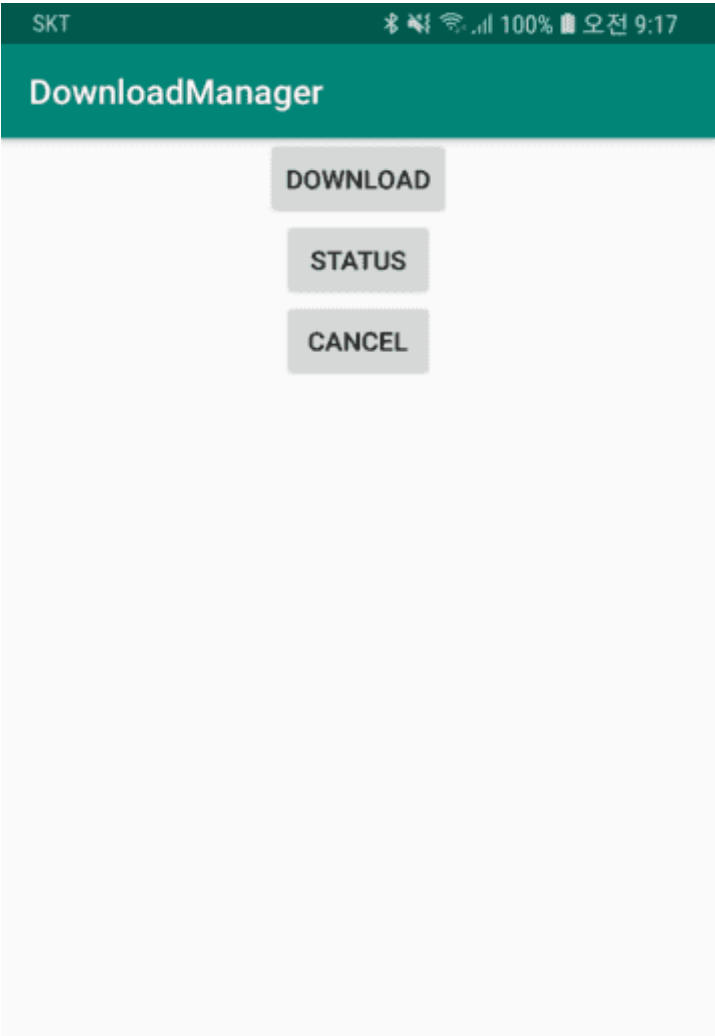
    <Button
        android:id="@+id/downloadBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Download"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

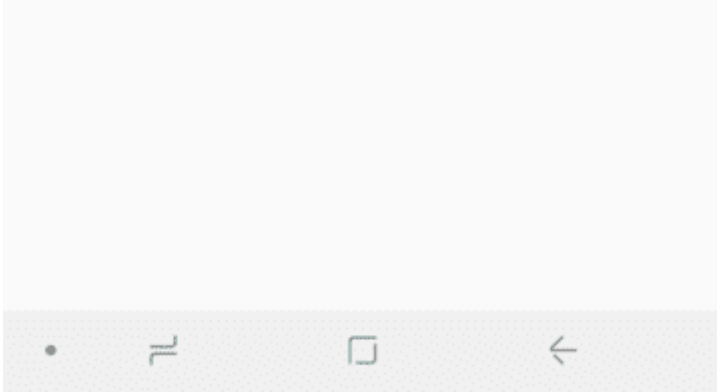
    <Button
        android:id="@+id/statusBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Status"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/downloadBtn"/>

    <Button
        android:id="@+id/cancelBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cancel"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/statusBtn"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

앱을 실행해보면 3개의 버튼을 갖고 있는 액티비티가 보입니다.





## 권한

인터넷에서 파일을 다운로드를 하기 때문에 **AndroidManifest.xml**에 인터넷 퍼미션을 꼭 추가해야합니다.

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

## 다운로드 요청(Request)

다운로드 버튼을 누르면 DownloadManager에 파일 다운로드를 요청하도록 구현하였습니다.

```
downloadBtn.setOnClickListener {
    loadImage()
}
```

다운로드 요청은 **DownloadManager.Request** 객체를 생성하고 **DownloadManager.enqueue**를 이용하여 추가만 해주면됩니다. **Request**객체에 noti피케이션 등에 대한 설정 및 다운로드에 대한 제약사항을 설정할 수 있습니다.

다운로드매니저 Queue에 Request가 추가되면 다운로드매니저의 백그라운드에서 알아서 파일을 다운로드합니다. 앱에서 별도로 백그라운드 쓰레드를 만들 필요가 없습니다.

**enqueue**는 downloadId를 리턴하는데요. 이 Id는 다운로드 상태를 알거나 결과를 확인하기 위해 필요합니다.

```
private fun loadImage() {
    val file = File(getExternalFilesDir(null), "dev_submit.mp4")
    val youtubeUrl = "https://r2---sn-oguelney.googlevideo.com/videoplayback?expire=1547361698&rat

    val request = DownloadManager.Request(Uri.parse(youtubeUrl))
        .setTitle("Downloading a video")
        .setDescription("Downloading Dev Summit")
        .setNotificationVisibility(DownloadManager.Request.VISIBILITY_VISIBLE)
        .setDestinationUri(Uri.fromFile(file))
        .setRequiresCharging(false)
        .setAllowedOverMetered(true)
        .setAllowedOverRoaming(true)

    downloadId = downloadManager.enqueue(request)
    Log.d(TAG, "path : " + file.path)
}
```

**Request**에 설정하는 항목은 다음과 같습니다.

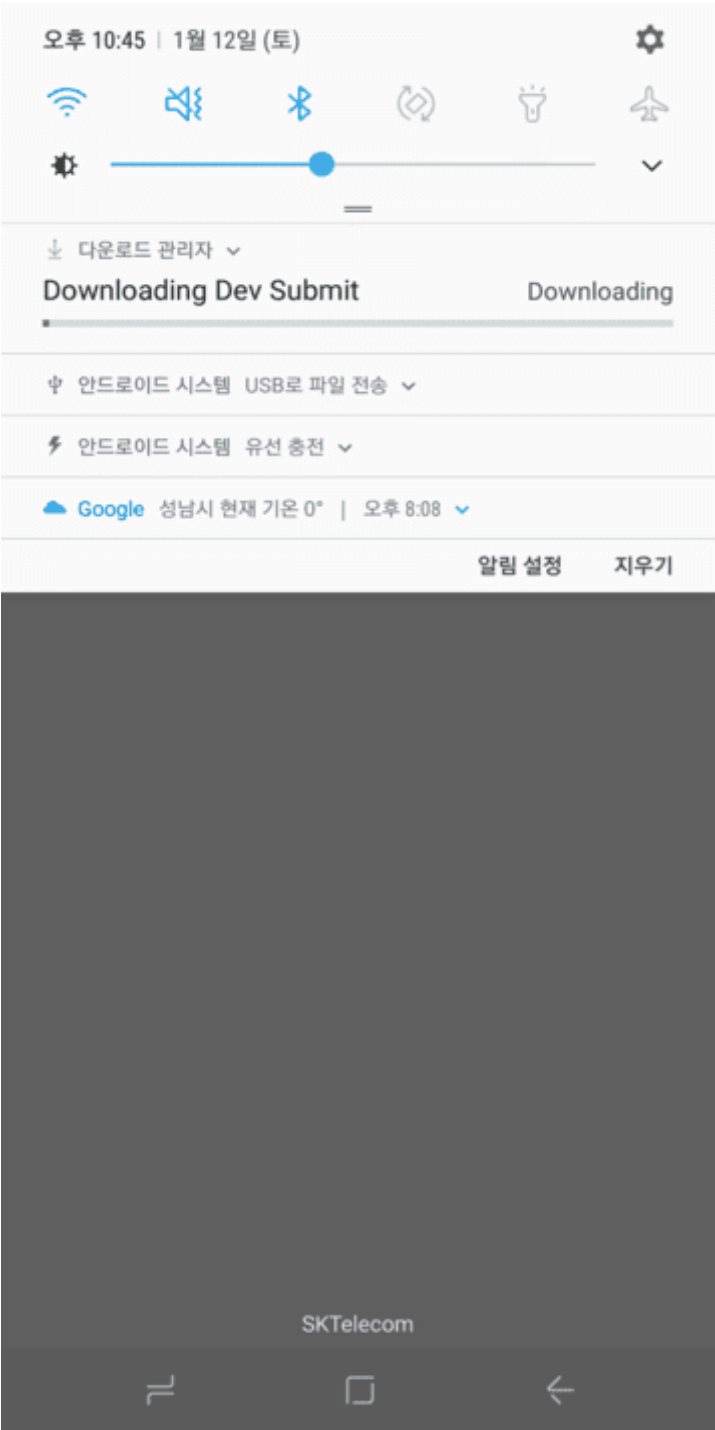
- **DownloadManager.Request:** Request 객체를 생성하며 인자로 다운로드할 파일의 URI를 전달합니다.
- **setTitle:** noti피케이션에 보이는 타이틀입니다.

- **setDescription:** noti피케이션에 보이는 디스크립션입니다.
- **setNotificationVisibility:** VISIBILITY\_VISIBLE로 설정되면 noti피케이션에 보여집니다.
- **setDestinationUri:** 파일이 저장될 위치의 URI입니다.
- **setRequiresCharging:** True로 설정되면, 단말이 충전중일 때만 다운로드합니다.
- **setAllowedOverMetered:** True로 설정되면, 모바일네트워크가 연결되었을 때도 다운로드합니다.
- **setAllowedOverRoaming:** True로 설정되면, 로밍네트워크가 연결되었을 때도 다운로드합니다.

setNotificationVisibility에 설정하는 옵션으로 아래와 같은 것들이 있습니다.

- **VISIBILITY\_VISIBLE:** 다운로드가 진행중일 때만 noti를 보여주며, 완료되면 보여주지 않습니다.
- **VISIBILITY\_VISIBLE\_NOTIFY\_COMPLETED:** 다운로드 진행 중 그리고 완료되었을 때 모두 noti를 보여줍니다.
- **VISIBILITY\_VISIBLE\_NOTIFY\_ONLY\_COMPLETION:** 다운로드가 완료되었을 때만 noti를 보여줍니다.
- **VISIBILITY\_HIDDEN:** noti를 보여주지 않습니다.

다운로드가 진행되면 noti피케이션으로 진행상황을 알려줍니다. 이 부분은 DownloadManager에 구현되어 있어서 따로 구현할 필요가 없습니다.



## 다운로드 상태 확인(Status)

상태 버튼을 누르면 다운로드 상태를 토스트로 출력하도록 하였습니다.

```
statusBtn.setOnClickListener {
    val status = getStatus(downloadId)
    Toast.makeText(this, status, Toast.LENGTH_SHORT).show()
}
```

다운로드 ID만 있으면 DownloadManager에 쿼리할 수 있습니다. **DownloadManager.Query** 객체를 생성하고 **DownloadManager.query** API로 쿼리를 합니다. 리턴 값으로 cursor가 리턴되며 특정 칼럼을 조회하여 다운로드 상태를 가져올 수 있습니다.

```
private fun getStatus(id: Long): String {
    val query: DownloadManager.Query = DownloadManager.Query()
    query.setFilterById(id)
    var cursor = downloadManager.query(query)
    if (!cursor.moveToFirst()) {
        Log.e(TAG, "Empty row")
        return "Wrong downloadId"
    }

    var columnIndex = cursor.getColumnIndex(DownloadManager.COLUMN_STATUS)
    var status = cursor.getInt(columnIndex)
    var columnReason = cursor.getColumnIndex(DownloadManager.COLUMN_REASON)
    var reason = cursor.getInt(columnReason)
    var statusText: String

    when (status) {
        DownloadManager.STATUS_SUCCESSFUL -> statusText = "Successful"
        DownloadManager.STATUS_FAILED -> {
            statusText = "Failed: $reason"
        }
        DownloadManager.STATUS_PENDING -> statusText = "Pending"
        DownloadManager.STATUS_RUNNING -> statusText = "Running"
        DownloadManager.STATUS_PAUSED-> {
            statusText = "Paused: $reason"
        }
        else -> statusText = "Unknown"
    }

    return statusText
}
```

다운로드 상태와 의미는 아래와 같습니다.

- **STATUS\_SUCCESSFUL**: 다운로드를 성공적으로 완료되었음
- **STATUS\_FAILED**: 다운로드가 실패되었음
- **STATUS\_RUNNING**: 현재 다운로드가 진행 중
- **STATUS\_PAUSED**: 다운로드가 중지되었고, 계속 받거나 다시 받기를 기다리는 상태

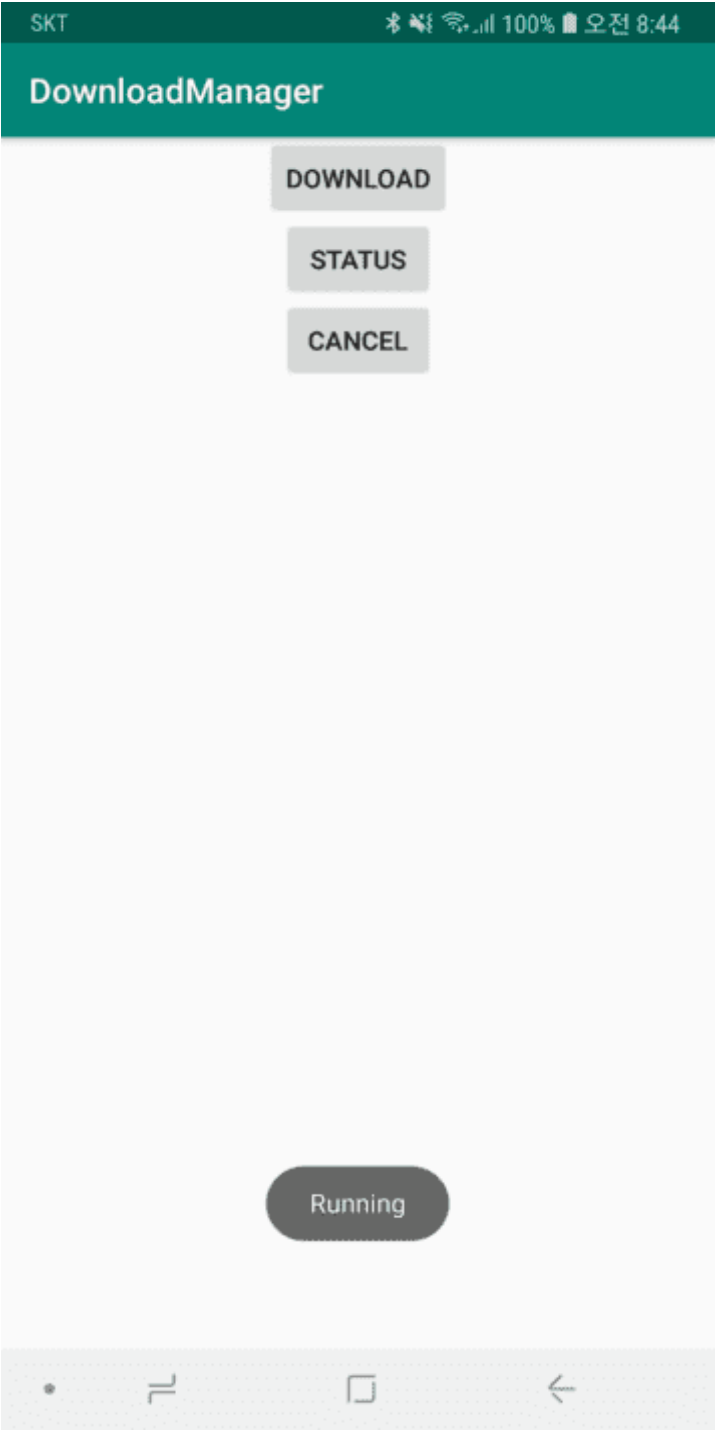
다운로드 상태가 **STATUS\_FAILED**, **STATUS\_PAUSED**일 때, 이렇게 된 이유(Reason)을 알 수 있습니다.

Reason의 종류와 아래와 같은 것들이 있습니다.

- PAUSED\_WAITING\_TO\_RETRY
- PAUSED\_WAITING\_FOR\_NETWORK
- PAUSED\_QUEUED\_FOR\_WIFI
- PAUSED\_UNKNOWN
- ERROR\_FILE\_ERROR
- ERROR\_UNHANDLED\_HTTP\_CODE
- ERROR\_HTTP\_DATA\_ERROR
- ERROR\_TOO\_MANY\_REDIRECTS
- ERROR\_TOO\_MANY\_REDIRECTS

- ERROR\_INSUFFICIENT\_SPACE
- ERROR\_DEVICE\_NOT\_FOUND
- ERROR\_CANNOT\_RESUME
- ERROR\_FILE\_ALREADY\_EXISTS
- ERROR\_UNKNOWN

앱을 실행하고 다운로드 버튼을 누른 뒤 Status버튼을 누르면 Running을 출력합니다.



## 다운로드 취소

다운로드 요청한 것에 대해서 취소도 할 수 있습니다. `DownloadManager.remove` API에 인자로 `downloadId`를 전달하면 됩니다.

취소버튼을 누르면 요청한 Id에 대해서 취소하도록 구현하였습니다.

```
cancelBtn.setOnClickListener {
    if (downloadId != -1L) {
        downloadManager.remove(downloadId)
    }
}
```

## 다운로드 결과 받기, noti 클릭 이벤트 받기

다운로드 요청이 완료되면 브로드캐스트로 결과를 전달해줍니다. 다운로드가 성공하거나 또는 실패해도 결과를 전달합니다.

사용자가 노티를 클릭하였을 때도 노티피케이션을 보내줍니다. 다운로드가 취소된 경우 다시 다운로드 받는 UI를 보여주거나, 다운받는 파일에 대한 자세한 정보를 보여주도록 구현할 수 있습니다.

먼저 DownloadManager에 다운로드를 요청하기 전에 브로드캐스트를 등록해야 합니다. DownloadManager는 다음 두 개의 [인텐트](#)를 전달할 수 있습니다.

- **ACTION\_DOWNLOAD\_COMPLETE**: 다운로드 요청이 완료(성공 또는 실패)되면 이 인텐트를 전달됩니다.
- **ACTION\_NOTIFICATION\_CLICKED**: 사용자가 노티피케이션을 클릭하면 이 인텐트가 전달됩니다.

동적으로 브로드캐스트 리시버를 등록하였습니다.

```
val intentFilter = IntentFilter()
intentFilter.addAction(DownloadManager.ACTION_DOWNLOAD_COMPLETE)
intentFilter.addAction(DownloadManager.ACTION_NOTIFICATION_CLICKED)
registerReceiver(onDownloadComplete, intentFilter)
```

브로드캐스트 리시버는 아래와 같이 구현할 수 있습니다.

```
private val onDownloadComplete = object : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val id = intent.getLongExtra(DownloadManager.EXTRA_DOWNLOAD_ID, -1)
        if (DownloadManager.ACTION_DOWNLOAD_COMPLETE.equals(intent.action)) {
            if (downloadId == id) {
                val query: DownloadManager.Query = DownloadManager.Query()
                query.setFilterById(id)
                var cursor = downloadManager.query(query)
                if (!cursor.moveToFirst()) {
                    return
                }

                var columnIndex = cursor.getColumnIndex(DownloadManager.COLUMN_STATUS)
                var status = cursor.getInt(columnIndex)
                if (status == DownloadManager.STATUS_SUCCESSFUL) {
                    Toast.makeText(context, "Download succeeded", Toast.LENGTH_SHORT).show()
                } else if (status == DownloadManager.STATUS_FAILED) {
                    Toast.makeText(context, "Download failed", Toast.LENGTH_SHORT).show()
                }
            }
        } else if (DownloadManager.ACTION_NOTIFICATION_CLICKED.equals(intent.action)) {
            Toast.makeText(context, "Notification clicked", Toast.LENGTH_SHORT).show()
        }
    }
}
```

인텐트는 여러 ID에 대한 응답이 올 수 있기 때문에 downloadId를 체크해야 합니다. 다운로드가 실패하는 경우에도 인텐트가 전달되기 때문에 상태를 체크해야 합니다.

## 참고

- 완성된 예제는 [GitHub](#)에 있습니다.
- <https://developer.android.com/reference/android/app/DownloadManager>

android

0 Comments

codechacha


Sungcheol Kim

Recommend

Tweet


Share

Sort by Best



Start the discussion...

Be the first to comment.

 Subscribe  Add Disqus to your siteAdd DisqusAdd  Disqus' Privacy PolicyPrivacy PolicyPrivacy PolicyPrivacy Policy

## Categories

- [Android](#)
- [Etc](#)
- [Java](#)
- [Kotlin](#)
- [Linux](#)
- [Python](#)

## Recent Posts

- [X-bows keyboard 구입 및 사용 후기](#)
- [안드로이드 boot.img의 커널과 램디스크를 수정하는 방법](#)
- [안드로이드 UIAutomator로 UI 테스트, 가장 쉽고 빠른 가이드](#)
- [삼성 안드로이드 오딘\(Odin\), Linux, Windows 실행 파일 다운로드](#)
- [안드로이드 - UsageStatsManager로 앱 실행 기록 가져오기](#)

## Related Posts

- [안드로이드 - RecyclerView 구현 방법 \(AndroidX, Kotlin\)](#)
- [안드로이드 앱이 32/64bit 기기에서 동작하도록 만들기](#)
- [Android App Bundle로 Apk 크기를 더 작게 만들기](#)
- [Android의 앱 데이터 폴더 경로 및 내부/외부 저장소 설명](#)
- [안드로이드 앱\(apk\)을 decompile하는 다양한 방법](#)