

# 배포 가이드

---

17.0.0.1000



이 문서에 잘못된 정보가 있을 수 있습니다. 투비소프트는 이 문서가 제공하는 정보의 정확성을 유지하기 위해 노력하고 특별한 언급 없이 이 문서를 지속적으로 변경하고 보완할 것입니다. 그러나 이 문서에 잘못된 정보가 포함되어 있지 않다는 것을 보증하지 않습니다. 이 문서에 기술된 정보로 인해 발생할 수 있는 직접적인 또는 간접적인 손해, 데이터, 프로그램, 기타 무형의 재산에 관한 손실, 사용 이익의 손실 등에 대해 비록 이와 같은 손해 가능성에 대해 사전에 알고 있었다고 해도 손해 배상 등 기타 책임을 지지 않습니다.

사용자는 본 문서를 구입하거나, 전자 문서로 내려 받거나, 사용을 시작함으로써, 여기에 명시된 내용을 이해하며, 이에 동의하는 것으로 간주합니다.

각 회사의 제품명을 포함한 각 상표는 각 개발사의 등록 상표이며 특허법과 저작권법 등에 의해 보호를 받고 있습니다. 따라서 본 문서에 포함된 기타 모든 제품들과 회사 이름은 각각 해당 소유주의 상표로서 참조용으로만 사용됩니다.

---

발행처 | (주)투비소프트

발행일 | 2018/07/23

주소 | (06083) 서울시 강남구 봉은사로 617 인탑스빌딩 2-5층

전화 | 02-2140-7700

홈페이지 | [www.tobesoft.com](http://www.tobesoft.com)

고객지원센터 | [support.tobesoft.co.kr](http://support.tobesoft.co.kr)

제품기술문의 | 1588-7895 (오전 10시부터 오후 5시까지)

# 변경 이력

---

버전	변경일	내용
17.0.0.100	2017-10-13	17.0.0.100 공개로 전환
17.0.0.200	2017-12-12	<a href="#">배포 개요</a> 항목에 변경된 내용 반영
17.0.0.300	2018-01-17	<a href="#">윈도우 런타임</a> 배포파일 설명에서 지원하지 않는 ActiveX DLL 항목을 삭제했습니다. 윈도우 <a href="#">Deploy</a> 설명 중에서 "unknown publisher"에 대한 설명 일부를 수정했습니다.
17.0.0.300	2018-01-19	<a href="#">앱 개발 및 실행 (iOS)</a> 항목 추가 <a href="#">앱 개발 및 실행 (안드로이드)</a> 항목 추가
17.0.0.300.3	2018-01-19	<a href="#">프로젝트 생성</a> 항목에 아이폰 X 관련 주의 문구를 추가했습니다.
17.0.0.500	2018-02-22	Deploy 기능 변경에 따라 일부 내용을 수정했습니다. <a href="#">Windows App</a>
17.0.0.700	2018-04-17	<a href="#">런처 서비스</a> 항목 추가
17.0.0.800	2018-05-24	Bluetooth 지원 추가 - <a href="#">iOS 라이브러리 및 프레임워크 설정</a> 에 CoreBluetooth.framework 추가 - <a href="#">AndroidManifest.xml</a> 에 Bluetooth 항목 추가 <a href="#">애플리케이션 설치 (사용자)</a> 시 Shortcut 생성 방식 변경 내용을 반영했습니다.
17.0.0.900	2018-06-19	<a href="#">MainActivity.java</a> 코드 수정 및 설명 추가 - <a href="#">NexacroResourceManager</a> 관련 코드 추가 - <a href="#">Update Type</a> 에 따른 설정부분 설명 추가
17.0.0.900.1	2018-06-25	iOS <a href="#">Config 설정</a> 항목 추가 Android <a href="#">Config 설정</a> 항목 추가
17.0.0.900.2	2018-07-04	<a href="#">앱 개발 및 실행 (macOS)</a> 항목 추가
17.0.0.1000	2018-07-23	<a href="#">Bootstrap URL</a> 적용 시 주의사항 항목 추가 <a href="#">Build configuration</a> 설정 시 오류 메시지 처리 항목 추가

# 차례

---

저작권 및 면책조항 .....	ii
변경 이력 .....	iii
차례 .....	iv

## 파트 I. 개요 ..... 1

1. 넥사크로플랫폼 배포 .....	2
1.1 개요 .....	2
1.2 배포 파일 .....	3
1.2.1 데스크탑 .....	3
윈도우 런타임 .....	3
macOS 런타임 .....	4
웹브라우저 .....	4
1.2.2 모바일 .....	5
안드로이드 런타임, iOS 런타임 .....	5

## 파트 II. Windows App ..... 6

2. 배포 개요 .....	7
2.1 Deploy .....	7
2.1.1 Engine Version .....	8
2.1.2 Bootstrap URL .....	8
2.1.3 운영체제 버전별 배포 .....	9
2.2 애플리케이션 설치 (사용자) .....	10

## 파트 III. 런처 서비스 ..... 11

3. 설치 및 기본 사용 .....	12
3.1 런처 서비스 설치 .....	12

3.1.1 설치 및 윈도우 서비스 등록 .....	12
3.1.2 런처 서비스 삭제 .....	13
3.1.3 특정 포트 지정 .....	14
3.2 동작 방식 .....	15
3.3 기본 설정 .....	16
3.3.1 주소 설정 .....	16
3.3.2 기본 요소 설정 .....	16
platform .....	16
action .....	16
id .....	17
value .....	17
3.3.3 데이터 전송 .....	18
3.3.4 통신 결과 처리 .....	18
<b>4. 지원기능 및 사용예제 .....</b>	<b>20</b>
4.1 지원기능 .....	20
4.1.1 속성 .....	20
4.1.2 메소드 .....	21
4.1.3 이벤트 .....	22
4.2 사용예제 .....	22
4.2.1 신규 id 요청 .....	22
4.2.2 필수 속성 설정 .....	23
4.2.3 메소드 실행 .....	24
4.2.4 전체 코드 .....	25
<b>파트 IV. App Builder (Android, iOS, macOS) .....</b>	<b>27</b>
<b>5. App 생성 .....</b>	<b>28</b>
5.1 Create App .....	28
5.2 App Info .....	28
5.2.1 General .....	29
5.2.2 Build configuration .....	30
5.2.3 Authority .....	30
5.2.4 Target OS .....	31
5.2.5 Nexacro application resource .....	32
5.3 Android platform .....	33
5.3.1 Build Configuration .....	33
5.3.2 Signing Info .....	35
5.3.3 User Library .....	35
5.3.4 Android Permission .....	35

5.4	iOS platform	36
5.4.1	Build configuration	37
5.4.2	Signing Info	38
5.4.3	User Library	38
5.5	macOS platform	39
5.5.1	Build configuration	39
5.5.2	Signing Info	40
5.6	Project Modification	40
5.7	Build App	41
6.	App 목록	44

## 파트 V. iOS/Android/macOS App 45

7.	앱 개발 및 실행 (iOS)	46
7.1	앱 개발 환경 설정	46
7.1.1	개발자 계정 등록	46
	Apple 계정 생성	46
	개발자 라이선스 인증	47
7.1.2	Xcode, iOS SDK 설치	48
7.1.3	배포에 필요한 작업	49
7.2	앱 프로젝트 개발	49
7.2.1	프로젝트 생성	50
7.2.2	iOS 라이브러리 및 프레임워크 설정	51
7.2.3	넥사코플랫폼 라이브러리 설정	53
7.2.4	리소스 설정	54
	이미지 설정	54
	메시지 설정	56
7.2.5	Config 설정	59
7.2.6	빌드 환경 설정	61
	AppDelegate.h	61
	AppDelegate.m	62
	main.m	62
	기타 설정	63
7.3	앱 테스트	64
8.	앱 개발 및 실행 (안드로이드)	66
8.1	앱 개발 환경 설정	66
8.1.1	JDK(Java SE Development Kit) 설치	66
	내려받기	66
	설치 확인	67

8.1.2	개발 도구 설치	67
8.2	앱 프로젝트 개발	68
8.2.1	프로젝트 생성	68
8.2.2	넥사크로플랫폼 라이브러리 설정	72
8.2.3	리소스 설정	75
	이미지 설정	76
	문자열 설정	78
	레이아웃 설정	81
8.2.4	Config 설정	81
8.2.5	빌드 환경 설정	84
	MainActivity.java	84
	AndroidManifest.xml	86
8.2.6	APK에 넥사크로 애플리케이션 임베딩(Asset)	91
8.3	빌드	92
8.3.1	앱 테스트	92
8.3.2	설치 파일 생성	94
9.	앱 개발 및 실행 (macOS)	95
9.1	준비 사항	95
9.1.1	개발 환경	95
9.2	앱 프로젝트 개발	95
9.2.1	프로젝트 생성	96
9.2.2	넥사크로플랫폼 라이브러리 설정	97
9.2.3	리소스 설정	98
9.2.4	빌드 환경 설정	98
	main.m	98
	기타 설정	99
부록 A.	경로설정	101
A.1	Alias 경로	101
A.2	nexacro.xml	102
A.3	상대경로	102
A.3.1	Project 내 경로의 상대경로 지원여부	102
부록 B.	예외상황	104
B.1	웹브라우저 옵션	104
B.1.1	자바스크립트 활성화	104
B.1.2	파일 다운로드 활성화	105
B.1.3	HTTP 1.1 활성화	105
B.1.4	XMLHTTP 활성화	105
B.2	인터넷 익스플로러 호환성 보기	106

B.3	기존 웹 화면에 아이프레임으로 콘텐츠 추가	107
-----	-------------------------	-----



파트 I.

---

개요

# 1.

## 넥사크로플랫폼 배포

사용자가 넥사크로플랫폼으로 개발된 애플리케이션을 사용하려면 사용 환경에 따라 적절한 실행 환경을 만들어주고 필요한 파일을 내려받아야 합니다. 이러한 과정을 배포라고 합니다. 배포된 파일은 캐시 정책에 따라 내부에서 관리되며 필요할 때 새로운 파일로 교체됩니다.

데스크탑, 모바일 환경에 이미 설치된 웹브라우저에서 넥사크로 애플리케이션을 실행하는 경우에는 웹서버에서 필요한 HTML, 자바스크립트, CSS 파일만 내려받아 바로 실행합니다. 하지만 런타임 방식에서는 별도의 배포, 설치 과정이 필요합니다. 세부적인 과정은 각 실행 환경에 따라 달라질 수 있습니다.

### 1.1 개요

넥사크로플랫폼 애플리케이션이 실행하기 위해 필요한 파일을 먼저 살펴보겠습니다. 실행 환경에 따라 배포되는 파일은 아래와 같습니다.

분류	설명	런타임	웹
넥사크로 브라우저	스크립트, 메모리, 렌더링 처리 런타임 엔진	O	X
넥사크로 프레임워크	프레임워크, 컴포넌트 라이브러리 (자바스크립트)	O	O
애플리케이션	애플리케이션 코드 (자바스크립트)	O	O

넥사크로 브라우저의 가장 큰 특징은 런타임 엔진입니다. 웹브라우저 환경에서는 필요한 프레임워크, 컴포넌트, 애플리케이션을 바로 사용하는 데 반해 넥사크로 브라우저를 사용하려면 각 실행 환경에 맞는 런타임 엔진을 별도로 배포 해주어야 합니다.

런타임 엔진을 추가로 배포하는 작업이 사용자로서도 불편할 수 있지만, 시스템 환경에 따라 기존에 사용하던 시스템과 연계가 필요하거나 웹브라우저에서 제공하지 못하는 확장된 기능을 사용하기 위해서는 넥사크로 브라우저를 선택할 수 있습니다.



넥사크로플랫폼에서 제공하는 기본적인 기능은 버전에 상관없이 같지만 디바이스 API나 외부 기기 연동 등 확장 기능은 실행 환경에 따라 다를 수 있습니다.

## 1.2 배포 파일

버전별 배포되는 파일을 상세하게 살펴보면 아래와 같습니다.



파일 위치 또는 서버 위치를 설명하는 Alias는 [Alias 경로](#)를 참고해주세요.



아래 설명된 파일명과 제공되는 라이브러리는 업데이트에 따라 변경될 수 있습니다.

### 1.2.1 데스크탑

#### 윈도우 런타임

넥사크로플랫폼 런타임 엔진 배포 후 실행 방식에 따라 애플리케이션을 호출합니다. 콘텐츠는 같지만 프레임워크 자바스크립트 파일을 서버에서 직접 받지 않고 dll 파일로 제공된 라이브러리에 포함해 배포합니다.

분류	파일	설명	파일 위치
Runtime Engine	v8.dll	V8 Script Engine Library	%nexacro%
	XMemLib.dll	Memory Management Library	%nexacro%
	XBasicLib.dll	Base Libaray	%nexacro%
	nexacrolib.dll	Platform & Render Engine Library	%nexacro%
	nexacro.exe	넥사크로플랫폼 Executor	%nexacro%
Framework DLL	XFrameworkLib.dll	Framework API Library	%nexacro%
Extend DLL Files	protocol adaptor dll	Network Protocol Adaptor	%UPDATE%
	External DLL	External DLLs	%UPDATE%
Runtime Cache	cache.db	Runtime Cache DB	%CACHE%
	cachedfiles	Runtime Cached Data Files	%CACHE%₩key_adl
Runtime Config Files	nexacro.xml	Runtime 실행 환경 파일 & UserProfile	%USERAPP%
Log Files	nexacro_xxxxx.log	로그 파일	%USERAPP%₩Log
Launcher	splash.png	Splash Image File	%USERAPP%

분류	파일	설명	파일 위치
Download Files	loadingimage.png	WaitCursor Image File	%USERAPP%
	nexacro.ico	Application Icon File	%USERAPP%
	globalvars.dat	Global Variable List 파일	%USERAPP%



배포되는 파일 목록은 엔진 버전이나 운영체제에 따라 달라질 수 있습니다.  
위의 표에서는 주요 파일 목록만 나열했습니다.

## macOS 런타임

macOS 런타임은 각 운영체제에 따라 별도 패키징 과정이 필요합니다. 넥사크로 스튜디오에서 생성된 애플리케이션 소스를 Archive 파일로 생성하고 각 운영체제에 맞게 배포 파일을 생성하게 됩니다.

분류	파일	파일 설명	Packing
macOS App Files	nexacro17.macOS.framework	iOS Library Framework	iOS App
	start_macos.json	iOS Archive Information	
	localizable.strings	iOS Locale String Information	
Execute HTML	index.html		Run Archive
Framework JS			Engine Archive
Component JS			Engine Archive
Resource Files	loadingimage.png		Engine Archive
Theme File	theme_*.css		Theme Archive
Application Files			Application Archive

## 웹브라우저

필요한 모든 파일을 웹브라우저에서 직접 로딩하는 방식으로 실행됩니다. 사용하는 웹브라우저에 따라 성능 차이가 있을 수 있습니다.

분류	파일	설명	서버 위치
Execute HTML	index.html		%WEBDEPLOY PROJECT%
Framework JS	BasicObjs.js		%WEBDEPLOY FRAMEWORK%
	SystemBase.js	System Utility	%WEBDEPLOY FRAMEWORK%
	Platform.js	Platform Objects	%WEBDEPLOY FRAMEWORK%
	CssObjs.js	Style Object	%WEBDEPLOY FRAMEWORK%
	ErrorDefine.js	Error Information	%WEBDEPLOY FRAMEWORK%
Component JS	CompBase.json	Component Base Module	%WEBDEPLOY COMPONENT%
	ComComp.json	Common Component Module	%WEBDEPLOY COMPONENT%
	Grid.json	Grid Component Module	%WEBDEPLOY COMPONENT%
	DeviceAPI.json	DeviceAPI Module	%WEBDEPLOY COMPONENT%

분류	파일	설명	서버 위치
Resource Files	waitimage.gif	WaitCursor Image File	%WEBDEPLOY RESOURCE%
Theme File	theme_*.css	Deploy 된 테마 소스	%WEBDEPLOY THEME%
Application Files		Deploy 된 애플리케이션 소스	%WEBDEPLOY PROJECT%

## 1.2.2 모바일

### 안드로이드 런타임, iOS 런타임

iOS/안드로이드 앱은 각 운영체제에 따라 별도 패키징 과정이 필요합니다. 넥사크로 스튜디오에서 생성된 애플리케이션 소스를 Archive 파일로 생성하고 각 운영체제에 맞게 배포 파일을 생성하게 됩니다.

분류	파일	파일 설명	Packing
Android App Files	nexacro17.jar	Android Library Jar	Android App
	libnexacro17.so	Android Library SO	
	star_android.json	Android Archive Information	
	strings.xml	Android Locale String Information	
iOS App Files	nexacro17.framework	iOS Library Framework	iOS App
	start_ios.json	iOS Archive Information	
	localizable.strings	iOS Locale String Information	
Execute HTML	index.html		Run Archive
Framework JS			Engine Archive
Component JS			Engine Archive
Resource Files	loadingimage.png		Engine Archive
Theme File	theme_*.css		Theme Archive
Application Files			Application Archive

**파트 II.**

---

**Windows App**

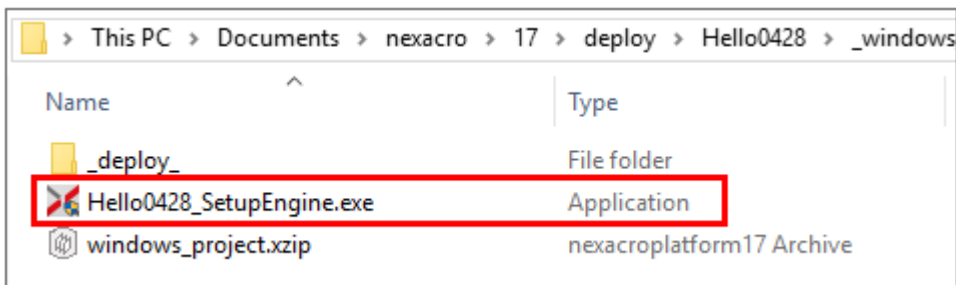
## 2.

## 배포 개요

넥사크로플랫폼으로 개발된 서비스를 윈도우 PC 사용자에게 제공하려면 넥사크로플랫폼 실행 환경을 사용자 PC에 설치해야 합니다. 마이크로소프트 오피스같은 소프트웨어를 설치하는 것과 같은 방식입니다. 넥사크로 스튜디오의 Deploy 메뉴를 사용해 설치 파일을 생성할 수 있으며 생성된 파일을 사용자에게 배포한 후 설치하도록 안내할 수 있습니다.

### 2.1 Deploy

넥사크로 스튜디오 메뉴[Deploy]에서 아카이브 파일을 생성하고 설치파일을 생성합니다. 설치 파일명이나 설치되는 경로를 수정할 수 있습니다. 설치 파일을 생성한 후 파일을 배포하고 사용자가 설치하도록 하는 과정은 시스템 운영 환경에 따라 달라질 수 있습니다.



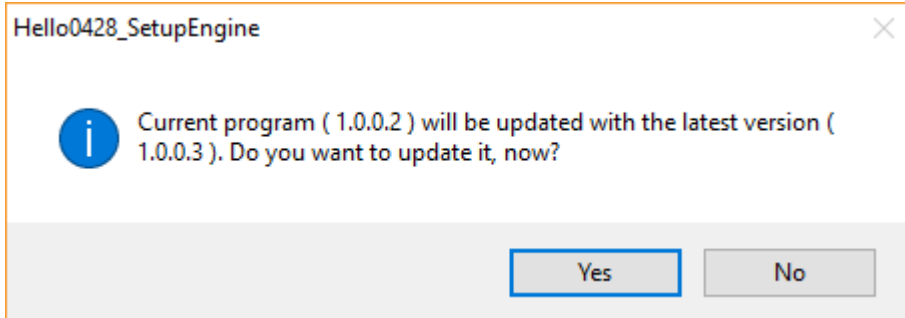
생성된 설치 파일을 실행 시 Publisher 정보는 "Unknown Publisher"로 표기됩니다.  
시스템 운영 환경에 따라 Publisher 정보가 확인되지 않은 경우 애플리케이션을 설치할 수 없는 경우도 있을 수 있습니다.



마이크로소프트에서 배포하는 SignTool.exe과 같은 도구를 사용해 디지털 서명을 처리할 수 있습니다.

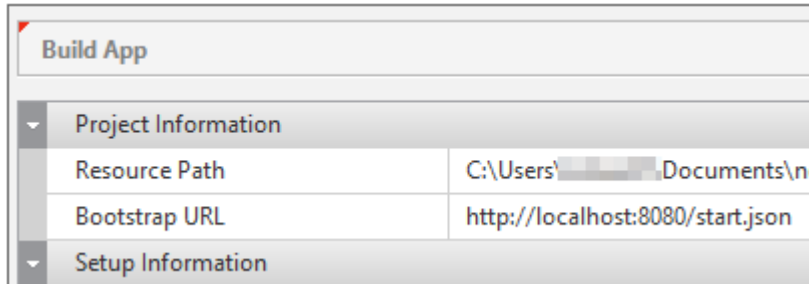
## 2.1.1 Engine Version

Engine Version 값을 변경한 경우에는 설치 파일 실행 시 변경된 버전을 확인한 후 설치를 진행할 수 있습니다.

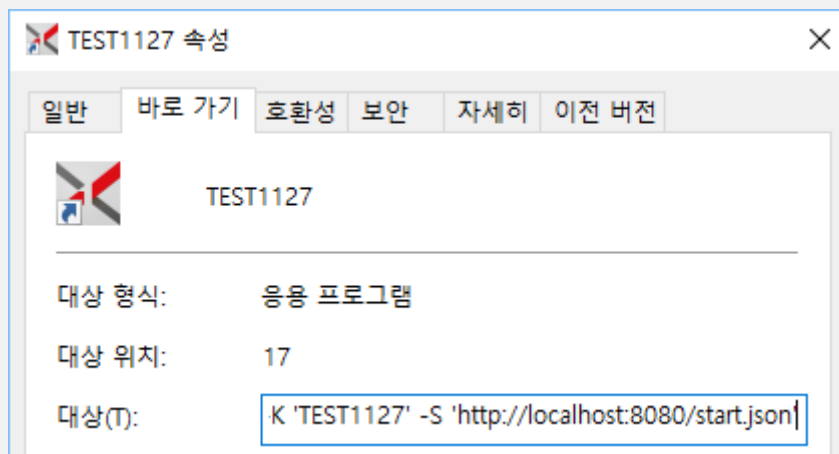


## 2.1.2 Bootstrap URL

사용자 PC에는 넥사크로플랫폼 엔진과 실행 및 업데이트에 필요한 정보만 설치하기 때문에 사용자 PC는 애플리케이션 실행에 필요한 콘텐츠를 가지고 있지 않습니다. 애플리케이션 실행을 위해서는 서버 URL을 "Bootstrap URL" 항목에 지정해주어야 합니다.



사용자에게 설치된 바로가기의 Bootstrap URL 속성값만 변경해서 사용하는 경우 설치된 제품 버전과 배포된 애플리케이션 버전이 달라 정상적으로 동작하지 않을 수 있습니다. 임의로 Bootstrap URL을 변경하는 것은 권장하지 않으며 다른 애플리케이션을 배포할 경우에는 Engine Setup Key, Install Location 값이 다른 설치파일을 생성하고 배포하는 것을 권장합니다.





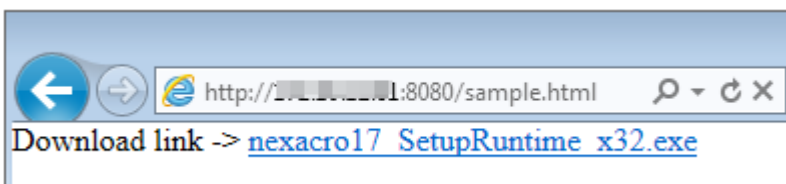
## 2.1.3 운영체제 버전별 배포

32비트, 64비트, XP 버전별 설치파일을 생성한 경우 사용자에게 모든 파일의 내려받기 정보를 제공한 후 사용자가 선택해서 내려받을 수도 있고 스크립트를 사용해 운영체제 버전을 확인 후 적절한 링크 정보를 제공할 수 있습니다.

```
<script type="text/javascript">
var SetupRuntimeName="";
function oninit()
{
    var userAgent = navigator.userAgent;
    if(userAgent.indexOf("WOW64") != -1
        || userAgent.indexOf("Win64") != -1)
        SetupRuntimeName="nexacro17_SetupRuntime_x64.exe";
    else if(userAgent.indexOf("Windows NT 5.1") != -1
        || userAgent.indexOf("Windows XP") != -1)
        SetupRuntimeName="nexacro17_SetupRuntime_XP.exe";
    else
        SetupRuntimeName="nexacro17_SetupRuntime_x32.exe";

    document.getElementById("test").innerHTML = "<a href='"+SetupRuntimeName+"'>"
        +SetupRuntimeName+"</a>"
}
</script>
<body onload="oninit()">
Download link -> <span id="test"></span>
</body>
```

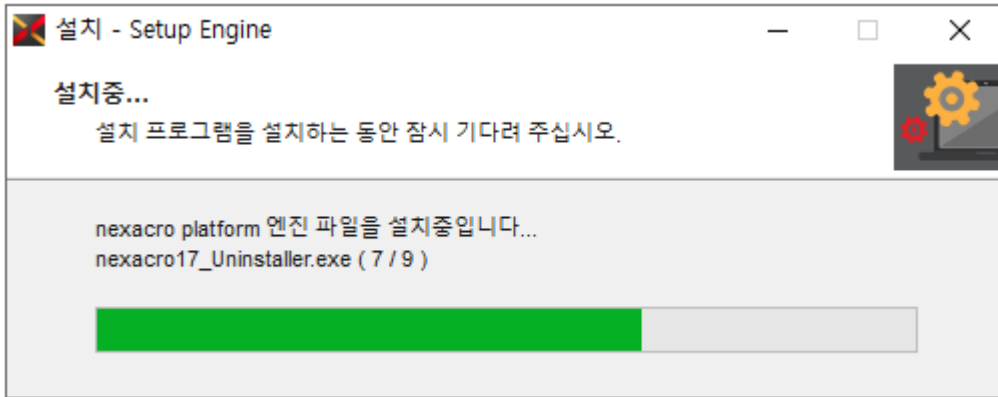
32비트 운영체제에서 해당 페이지 접근 시 운영체제 버전을 확인하고 적절한 링크를 제공할 수 있습니다.



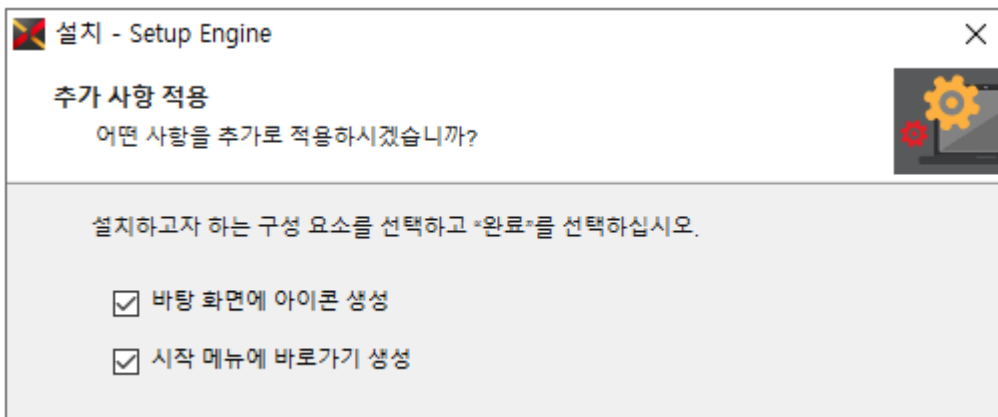
스크립트는 설명을 위해 작성된 코드입니다. 운영하는 시스템에 맞게 변경해 적용하실 수 있습니다.  
운영체제 변경에 따라 해당 스크립트가 동작하지 않을 수도 있습니다.

## 2.2 애플리케이션 설치 (사용자)

설치 파일을 실행하면 넥사크로플랫폼 엔진 설치 과정이 진행됩니다. 설치 폴더와 설치된 프로그램의 이름은 Setup Manager에서 변경할 수 있습니다.



설치가 완료되면 배포 시 설정에 따라 바로가기 요소를 설치할지 여부를 확인합니다. Shortcut Option 설정 시 자동 생성으로 설정된 경우에는 추가 확인 없이 바로 Shortcut을 생성합니다.



## 파트 III.

---

## 런처 서비스

## 3.

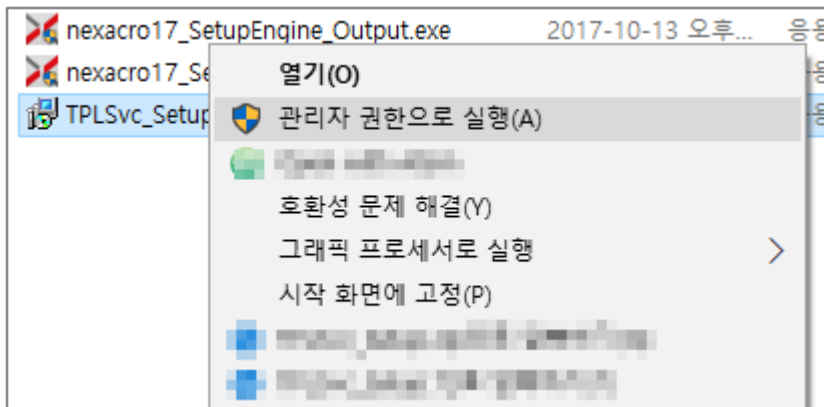
# 설치 및 기본 사용

## 3.1 런처 서비스 설치

### 3.1.1 설치 및 윈도우 서비스 등록

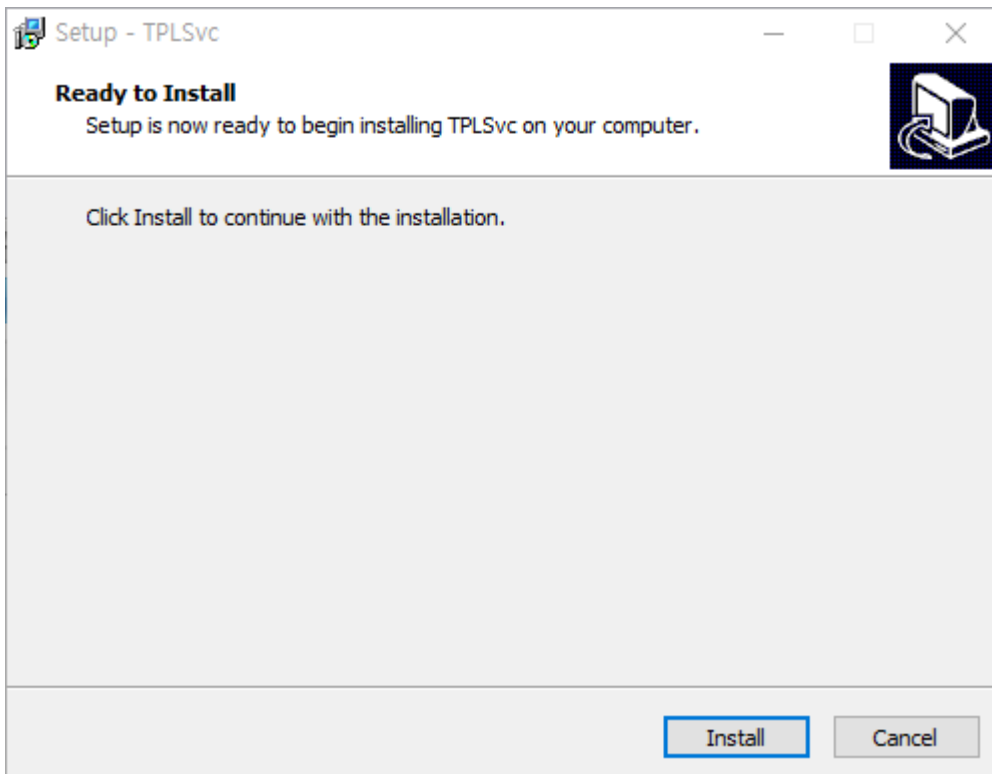
런처 서비스는 윈도우 서비스로 등록되어 동작합니다. 런처를 사용하기 위해서는 먼저 윈도우 서비스를 등록해야 합니다.

- ① 설치 파일을 내려받아 관리자 권한으로 실행합니다.

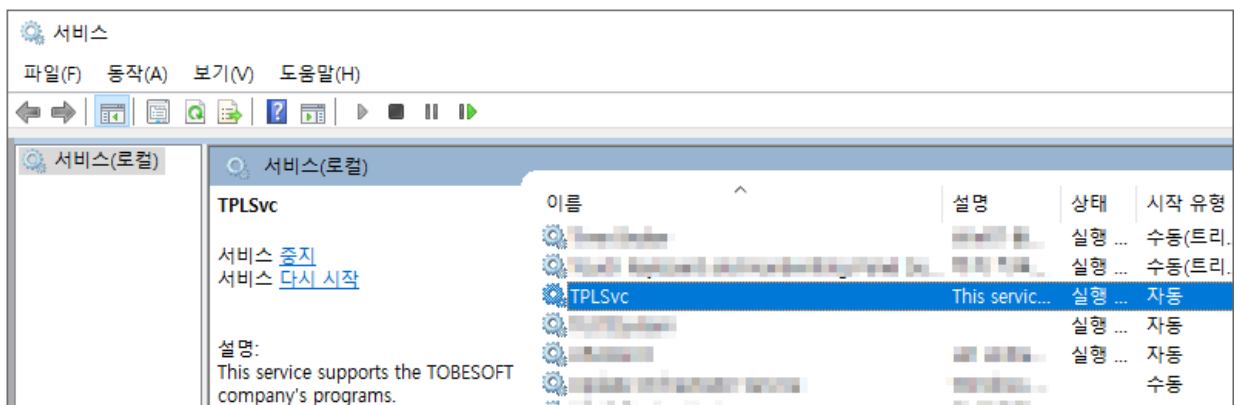


설치 시 관리자 권한으로 실행하지 않은 경우에는 서비스 동작 시 오류가 발생할 수 있습니다.

- ② 설치 마법사 화면이 실행됩니다. [InstalI] 버튼을 클릭하고 설치를 진행합니다. 설치가 완료되면 윈도우 서비스로 등록되고 자동으로 실행됩니다.



[제어판 > 시스템 및 보안 > 관리자 도구 > 서비스] 항목에서 등록된 서비스가 동작하는 것을 확인할 수 있습니다. 해당 창에서 서비스를 중지하거나 다시 시작할 수 있습니다.



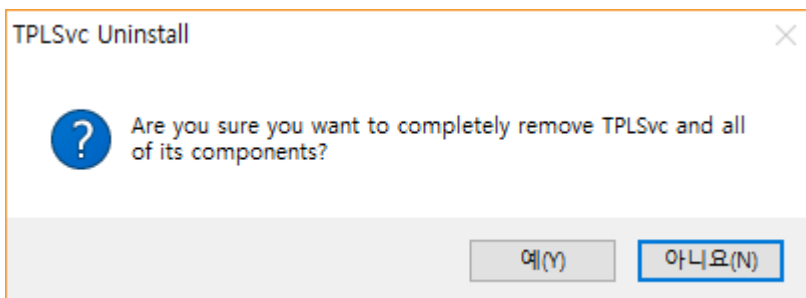
### 3.1.2 런처 서비스 삭제

- 1 설치된 앱(프로그램) 목록에서 TPLSvc 항목을 찾아 [제거] 버튼을 클릭합니다.

## 앱 및 기능



- ② 삭제 여부를 확인합니다. [예] 버튼을 클릭하면 삭제가 진행됩니다. 등록된 서비스가 해제되고 런처 서비스가 삭제됩니다.



### 3.1.3 특정 포트 지정

런처 서비스는 사용하는 포트가 지정되어 있는데, 이를 변경하고자 하는 경우에는 명령 프롬프트에서 직접 파라미터를 지정해 서비스를 실행할 수 있습니다.

- ① 런처 서비스가 설치된 경로를 확인합니다. 아래 경로에서 설치된 실행 파일을 확인할 수 있습니다.

```
C:\Program Files (x86)\TPLSvc\TPLSvc.exe
```

- ② 관리자 권한으로 명령 프롬프트를 실행하고 아래와 같이 파라미터를 지정해 윈도우 서비스를 해제할 수 있습니다. 런처 서비스를 삭제하는 것이 아니라 등록된 서비스만 해제합니다.

```
TPLSvc_Setup.exe -r false
```

- ③ 서비스를 다시 등록하기 위해서는 아래와 같이 실행합니다.

TPLSvc\_Setup.exe -r true

런처는 7895~7935 대역 내 포트 중 하나를 사용합니다. 런처가 실행되는 시스템에 다른 서비스가 해당 포트를 이미 사용하고 있다면 포트를 변경해 사용할 수 있습니다. 이런 경우에는 서비스 등록 시 포트를 지정하는 파라미터(-p 포트번호1, -p 포트번호2 ... -p 포트번호n)를 추가로 지정합니다.

예를 들어 8080포트와 8888포트를 사용하고자 한다면 아래와 같이 서비스를 등록합니다.

TPLSvc\_Setup.exe -r true -p 8080 -p 8888

```

C:\Program Files (x86)\TPLSvc>TPLSvc.exe -r false
Service stopped successfully

C:\Program Files (x86)\TPLSvc>TPLSvc.exe -r true -p 8080 -p 8888
Service start pending...
Service started successfully.
  
```

## 3.2 동작 방식

런처 서비스와 웹 애플리케이션 간 동작 방식을 간단하게 설명하면 아래 표와 같습니다.

런처 서비스		웹 애플리케이션
(2) id 할당 및 저장 후 id 반환	↔	(1) 신규 id 요청 (create)
(4) 필수 속성 저장 후 결과값 반환	↔	(3) 필수 속성 설정 (setproperty)
메소드 실행 후 결과값 반환	↔	메소드 실행 요청 (method)
속성 정보 반환	↔	속성 정보 요청 (getproperty)
이벤트 정보 반환	↔	이벤트 정보 요청 (event)
id에 해당하는 정보삭제	↔	id에 해당하는 정보삭제요청 (destroy)

- 웹 애플리케이션에서 런처 서비스 실행을 위해 형식에 맞는 데이터를 전송해야 합니다.
- 데이터는 JSON 형식으로 작성된 텍스트로 전송하며 결과값도 같은 형식입니다.
- id 값은 런처 실행을 위한 필수 정보입니다. 반드시 신규 id 요청을 전송해야 합니다. (1, 2번 동작)
- 필수 속성 정보를 설정해야 합니다. (3, 4번 동작)
- 대소문자를 구분하므로 주의해야 합니다.

## 3.3 기본 설정

### 3.3.1 주소 설정

주소 설정은 필수 부분과 추가 부분으로 구분할 수 있습니다. 필수 부분에 추가 부분을 조합해 사용합니다.

필수 부분	로컬주소, 포트, 예약어(launcher)
추가 부분	플랫폼명, 타임스탬프

`http://127.0.0.1:7895/launcher/nexacro/123456789`

`http://127.0.0.1:7895/launcher/xplatform/123456789`

`http://127.0.0.1:7895/launcher/miplatform/123456789`

- 로컬주소(127.0.0.1)과 예약어(launcher)은 수정할 수 없습니다.
- 설치 시 특정 포트를 열어놓은 경우에는 해당 포트 번호를 사용할 수 있습니다.
- 타임스탬프는 웹브라우저 캐시 동작으로 통신이 되지 않는 것을 방지합니다.

### 3.3.2 기본 요소 설정

기본적으로 설정하는 요소는 platform, action, id, value입니다. 각 용도에 맞게 설정합니다.

#### platform

플랫폼을 나타내는 문자열입니다. 제품에 따라 아래 3가지 중 하나를 지정할 수 있습니다.

`nexacro`, `xplatform`, `miplatform`

#### action

런처 서비스에 요청할 동작 문자열을 지정합니다. 아래와 같은 동작을 지정할 수 있습니다.

동작	설명
create	런처 서비스에서 최초 id를 할당받을 때
destroy	할당받은 id에 해당하는 정보를 런처 서비스에서 삭제
setproperty	속성 설정
getproperty	속성 정보 확인
method	메소드 실행
event	이벤트 관련 정보 확인



## id

런처 서비스로부터 할당받은 문자열이며 반드시 최초 동작 시 요청해야 합니다.

## value

동작에 대한 구체적인 JSON 형식의 정보이며 동작에 따라 구성이 다릅니다.

최초 id를 할당받기 위한 JSON : value 없음, 결과로 id가 전달됨

```
var objNexacro = new Object();
objNexacro.platform = 'nexacro';
objNexacro.action = 'create';
//objXP.id = ''; // 통신이 성공하면 정보가 채워지므로 생략해도 무관합니다.
```

관련 속성 설정을 위한 JSON

```
var objNexacro = new Object();
objNexacro.platform = 'nexacro';
objNexacro.action = 'setproperty';
objNexacro.value = {"property name" : property value, ... };
```

설정된 속성값을 확인하기 위한 JSON

```
var objNexacro = new Object();
objNexacro.platform = 'nexacro';
objNexacro.action = 'getproperty';
objNexacro.value = {"property name" : "", ... };
```

메소드 호출을 위한 JSON

```
var objNexacro = new Object();
objNexacro.platform = 'nexacro';
objNexacro.action = 'method';
objNexacro.value = {"method name": {"param": method parameter array, "result": }};
```

이벤트 정보를 확인하기 위한 JSON

```
var objNexacro = new Object();
objNexacro.platform = 'nexacro';
objNexacro.action = 'event';
objNexacro.value = {"event name" : {...}};
```

### 3.3.3 데이터 전송

데이터 전송 방식은 GET 방식과 POST 방식으로 구분합니다.

GET 방식은 주소 뒤에 ?와 JSON 형식의 문자열이 첨부됩니다

```
var objNexacro = new Object();
var xhrObject = new XMLHttpRequest();
xhrObject.onreadystatechange = resultProcess;
var jsonData = JSON.stringify(objNexacro);
var openurl = "http://127.0.0.1/launcher/nexacro/"+new Date().getTime();
xhrObject.open("GET", openurl+"?" +jsonData, "true");
xhrObject.send(null);
```

POST 방식은 send 메소드 파라미터에 JSON 형식의 문자열이 첨부됩니다

```
var objNexacro = new Object();
var xhrObject = new XMLHttpRequest();
xhrObject.onreadystatechange = resultProcess;
var jsonData = JSON.stringify(objNexacro);
var openurl = "http://127.0.0.1:80/launcher/nexacro/"+new Date().getTime();
xhrObject.open("POST", openurl, "true");
xhrObject.send(jsonData);
```

데이터 전송은 JSON 형식의 문자열을 구성하면 어떤 식으로든 처리할 수 있습니다. stringify 메소드를 지원할 수 없거나 직접 설정하기 원한다면 다음과 같은 형태로 문자열을 만들어 처리합니다.

```
create
'{"platform":"nexacro","action":"create"}'
```

launch method

id값은 create에서 받은 값으로 치환해야 합니다.

```
'{"platform":"nexacro","id":"12345678","action":"method", "value":{"launch":
null }}'
```

### 3.3.4 통신 결과 처리

JSON 형식의 문자열 데이터를 적절히 변환해 처리합니다.

```
var result = xhrObject.responseText; // xhrObject는 XMLHttpRequest임
var objResult = eval('(' + result + ')');
```

처리결과가 성공이라면 result 문자열이 'success' 로 처리됩니다.

```
if(objResult.result == "success" && objResult.id.length > 0)
{
    objNexacro = objResult; // objXP는 송신정보를 담고 있었던 객체임
} else {
    window.console.log("error!!!");
}
```

## 4.

# 지원기능 및 사용예제

## 4.1 지원기능

### 4.1.1 속성

Property	default	Data Type	설명
key	없음	string	<b>필수속성입니다.</b> 서비스의 Key값을 설정. 동일 start.json 상의 유일한 값이어야 함.
bjson	없음	string	<b>필수속성입니다.</b> 서비스의 start.json 경로를 지정.
globalvalue	없음	string	넥사크로플랫폼 Engine이 구동될 때 globalvariable에 추가될 변수를 설정.
splashimage	없음	string	넥사크로플랫폼 Engine이 Loading되는 동안 보여줄 스플래시 이미지 경로를 지정. 지정하지 않을 경우 넥사크로플랫폼 기본이미지가 출력되며, 이미지는 화면의 중앙에 표시됨.
onlyone	false	boolean	key와 bjson 값에 매칭되는 Instance를 하나만 띄울 것인지에 대한 여부를 설정.
enginesetupkey	없음	string	실행경로로 사용할 특정 버전의 넥사크로플랫폼 ProductKey 값을 지정. 사이트에서 별도 작성한 Setup Module을 사용하는 경우, 이 Property에 Setup을 만들 때 부여되는 Product Key를 설정함. 설정하지 않을 경우 서비스가 제대로 동작하지 않을 수 있음.

## 4.1.2 메소드

Method	Result	Parameter	Description
launch	없음	없음	넥사크로플랫폼 Engine을 넥사크로 브라우저로 실행시킴.
makeshortcut	없음	strShortCutName strIconPath strPosition bAllUser	<p>단축아이콘을 생성함.</p> <ul style="list-style-type: none"> <li>- strShortCutName:바로가기 아이콘명</li> <li>- strIconPath: 바로가기 아이콘의 경로</li> <li>- strPosition: 바로가기 생성위치 "startmenu"/ "startup" / "programs" / "desktop" 중 택일함 잘못된 값이 들어오면 "desktop"으로 처리함</li> <li>- bAllUser :전체 사용자가 사용할 수 있도록 생성할 지에 대한 여부(default는 false)</li> </ul>
IsExistShortcut	boolean	strShortcutName strPosition bAllUser	<p>입력받은 인자에 해당하는 바로가기 아이콘이 존재하는지 확인.</p> <ul style="list-style-type: none"> <li>- strShortCutName:바로가기 아이콘명</li> <li>- strPosition: 바로가기 생성위치 "startmenu"/ "startup" / "programs" / "desktop" 중 택일함 잘못된 값이 들어오면 "desktop"으로 처리함</li> <li>- bAllUser :전체 사용자가 사용할 수 있도록 생성할 지에 대한 여부(default는 false)</li> </ul>
getengineversion	string	strEngineKeyName	<p>시스템에 설치된 엔진 키에 해당하는 엔진 설치 버전을 가져옴.</p> <p>strEngineKeyName: Engine Setup Key</p>
download	없음		Application 실행 시 사용되는 자원을 다운로드.
deleteshortcut	boolean	strShortcutName strPosition bAllUser	<p>바로가기 아이콘을 삭제.</p> <ul style="list-style-type: none"> <li>- strShortCutName:바로가기 아이콘명</li> <li>- strPosition: 바로가기 생성위치 "startmenu"/ "startup" / "programs" / "desktop" 중 택일함 잘못된 값이 들어오면 "desktop"으로 처리함</li> <li>- bAllUser :전체 사용자가 사용할 수 있도록 생성할 지에 대한 여부(default는 false)</li> </ul>
addWebInfo	없음	strCookie	<p>현재 브라우저의 쿠키정보를 자동으로 globalvariable의 cookie변수에 추가.</p> <p>strCookie: 브라우저의 쿠키정보</p>

### 4.1.3 이벤트

Event	parameter	Occurs
error	nError strMsg	런처 서비스 실행 중에 에러가 발생하면 세팅되는 이벤트
		- nError: 에러코드 - strMsg: 에러 메시지

## 4.2 사용예제

웹 애플리케이션을 어떻게 구현하는지 간략한 예제를 설명합니다. 예제는 자바스크립트로 구현되었으며 사용하는 웹 애플리케이션에 맞게 수정해 사용할 수 있습니다.



아래 예제는 구글 크롬 브라우저에서 넥사크로플랫폼 애플리케이션을 동작하는 예제입니다. IE 브라우저에서 동작하는 예제는 전체 코드를 참조해주세요.

### 4.2.1 신규 id 요청

platform, action 요소 항목값을 채우고 신규 id를 요청합니다.

```
function start()
{
    objNexacro.platform = 'nexacro';
    objNexacro.action = 'create';
    sendData(true, true, createProcess);
}

function sendData(openpost, is_create, resultCallback)
{
    delete xhrCreateObject;
    xhrCreateObject = null;

    var sendObj = null;
    sendObj = new XMLHttpRequest();
```

```

sendObj.reqType = 1;

xhrCreateObject = sendObj;
sendObj.onreadystatechange = resultCallback;
var jsonData;
jsonData = JSON.stringify(objNexacro);
var timestamp = "/" + new Date().getTime();
var send_url = openurl + ":" + openport + openurl_add + timestamp;
console.log(send_url);

sendObj.open("POST", send_url, "true");
sendObj.send(jsonData);
console.log(jsonData);
return sendObj;
}

function createProcess()
{
    if ( xhrCreateObject.readyState == 4 || xhrCreateObject.reqType == 2) {
        console.log(xhrCreateObject.responseText);
    }
}

```

요청되는 jsonData 항목은 아래와 같습니다. 요소 항목만 채워서 전송됩니다.

```

{"platform":"nexacro","action":"create"}

```

런처 서비스에서 요청을 받고 id를 할당해 반환합니다. id 항목과 result 항목을 확인합니다. id값은 이후 요청 시 활용하게 됩니다.

```

{"action":"create","id":"1522821857","platform":"nexacro","result":"success"}

```

## 4.2.2 필수 속성 설정

예제에서 [property setting(basic)] 버튼 클릭 시 동작하는 기능입니다. 속성 중에서 key, bson, enginesetupkey 항목값을 채웁니다.

```

function do_property(test_action, action_sub)
{

```

```
objNexacro.action = 'setproperty';
objNexacro.value = { "key": "LauncherService", "bjson": "http://127.0.0.1:8080/start.json",
"enginesetupkey":enginesetupkey};
sendData(true, false, resultProcess);
}
```

넥사크로플랫폼 애플리케이션 배포 시 Local 설정으로 배포한 경우에는 start.json 경로를 로컬 주소로 지정해야 합니다. 이런 경우에는 아래와 같이 지정합니다.

```
function do_property(test_action, action_sub)
{
    objNexacro.action = 'setproperty';
    objNexacro.value = { "key": "LauncherService", "bjson": "D:\\nexacro\\17\\start.json", "
enginesetupkey":enginesetupkey};
    sendData(true, false, resultProcess);
}
```

요청되는 jsonData 항목은 아래와 같습니다. 이전 단계에서 확인된 id값을 설정하고 key, bjson, enginesetupkey 항목값이 채워져 있습니다.

```
{"action":"setproperty","id":"1522822903","platform":"nexacro","result":null,"value":{"key":"
LauncherService","bjson":"http://127.0.0.1:8080/start.json","enginesetupkey":"{94FAAFF5-0E
54-4539-AA67-*****}"}
```

런처 서비스에서 필수 속성값을 저장하고 성공 여부("result":"success")를 반환합니다.

```
{"action":"setproperty","id":"1522823068","platform":"nexacro","result":"success","value":{"
bjson":"http://127.0.0.1:8080/start.json","enginesetupkey":"{94FAAFF5-0E54-4539-AA
67-*****}","key":"LauncherService"}}
```

### 4.2.3 메소드 실행

예제에서 [launch] 버튼 클릭 시 동작하는 기능입니다.

```
function do_method(test_action) {
    objNexacro.action = 'method';
    objNexacro.value = '{ "launch": null }';
    sendData(true, false, resultProcess);
}
```



이전 단계에서 key, bjson, enginesetupkey 항목값을 전송했고 런처 서비스에서 해당하는 값을 저장하고 있습니다. id값을 키값으로 값을 저장하고 있기 때문에 이후 메소드를 실행하거나 다른 동작을 할때는 id값을 기준으로 동작하게 됩니다.

```
{"action":"method","id":"1522823068","platform":"nexacro","result":null,"value":{"launch":null}}
```

런처 서비스에서 메소드를 실행하고 성공 여부("result":"success")를 반환합니다. 정상적으로 메소드가 실행되었다면 운영체제에 설치된 애플리케이션이 실행됩니다.

```
{"action":"method","id":"1522823068","platform":"nexacro","result":"success","value":{"launch":null}}
```

## 4.2.4 전체 코드

아래 링크에서 사용 예제 코드를 내려받을 수 있습니다.

[localtest\\_nexacro\\_sample.html](#)

예제 코드 내에는 아래와 같은 문제 처리 방법이 포함되어 있습니다.

- 인터넷 익스플로러(IE) 8, 9 버전에서 Cross Domain 문제 회피 방법  
XMLHttpRequest.open 메소드 사용 시 액세스를 허용하지 않는다는 에러 메시지가 나올 경우 XMLHttpRequest 객체 대신 XDomainRequest 객체를 사용하면 에러가 발생하지 않습니다.

```
if (window.XMLHttpRequest) {
    sendObj = new XMLHttpRequest();
    sendObj.reqType = 1;
} else if (window.XDomainRequest) {
    sendObj = new XDomainRequest();
    sendObj.reqType = 2;
}
```

- 런처 서비스에서 열린 포트를 검색한 후 사용하는 방법  
모든 경우에 사용해야 하므로 action이 'create'인 경우에 지속적으로 통신시도를 해 결과값으로 id를 받았을 때 port 번호를 이후 통신에 사용하는 방법입니다.

```
function http_onerror()
{
    if (objNexacro.action == 'create') {
```

```
    if (findport == false) {
        if ( setport > 0) {
            setport = 0;
            openport = parseInt(getCookie("tplsvcopenport"), 10) | 0;
            if (openport > 0) {
                sendData(true, true, createProcess);
                return;
            }
        }
        openport = 7895;
    } else {
        openport++;
    }
    findport = true;

    if (openport <= 7935) //신규 런처는 7895 ~ 7935 사이 하나의 port를 open 함;
        sendData(true, true, createProcess);
    else
        alert("런처 서비스가 설치되지 않았거나 동작이 멈춰있는지 확인 바랍니다.");
}
}
```

## 파트 IV.

---

### App Builder (Android, iOS, macOS)

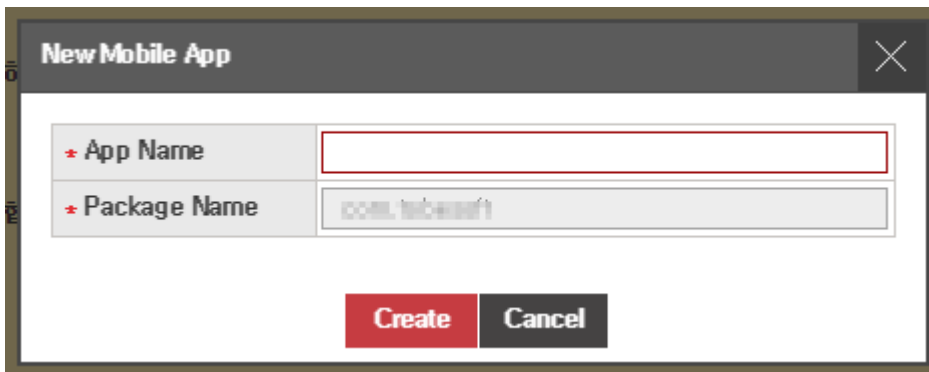
## 5.

# App 생성

App Builder에서 앱을 빌드하기 위한 기본 단위는 "App"입니다. 새로운 앱을 빌드하기 위해서는 "App"을 생성해야 하며 생성된 "App"을 수정하거나 삭제할 수 있습니다. "App" 설정에 따라 Android, iOS, macOS 앱을 빌드하고 배포할 수 있습니다.

## 5.1 Create App

App Builder 메뉴[App > Create App] 항목을 선택하면 새로운 "App"을 생성할 수 있는 창이 나타납니다. 생성할 "App" 이름을 지정하고 새로운 "App"을 생성합니다. "App" 이름은 생성된 이후 수정할 수 없습니다. App Builder 내부에서 "App" 목록 관리를 위한 이름일 뿐 실제 앱 빌드에 영향을 미치지 않습니다.



The screenshot shows a 'New Mobile App' dialog box. It has a title bar with the text 'New Mobile App' and a close button (X). Below the title bar, there are two input fields. The first field is labeled 'App Name' with a red asterisk, and it is currently empty. The second field is labeled 'Package Name' with a red asterisk, and it contains the text 'com.tobiasoft'. At the bottom of the dialog, there are two buttons: 'Create' (in red) and 'Cancel' (in dark grey).

## 5.2 App Info

[Create] 버튼을 클릭하면 "App Info" 항목을 입력할 수 있는 화면으로 전환됩니다. 정보를 입력하고 [Save] 버튼을 클릭하기 전에는 "App"이 생성되지 않습니다.

mobile0712

Save View List Refresh

App Info Android platform iOS platform macOS platform Build Application

**1 General**

- Package Name: com.nexacro0712
- Version: 1.0.0
- App Title: TEST0213

**2 Build configuration**

- Nexacro Project URL: http://c.../appbuilder/archives/283
- Build mode: ☐ debug ☒ release
- Detail Error Message Output: ☐ true ☒ false
- Nexacro Mobile Library: lib\_20180717\_0

**3 Authority**

- Owner: 11011002
- Access: ☐ share ☐ public ☒ private

**4 Target OS**

- ☒ Android: start\_android.isn (Upload detail)
- ☒ iOS: start\_ios.isn (Upload detail)
- ☒ macOS: start\_macos.isn (Upload detail)

**5 Nexacro application resource**

os	name	version	upload
ALL	nexacro17lib.zip	0	upload
ALL	Archive00.xc3e	0	upload
ALL	Resource.zip	0	upload
android	mobile0712.apk	0	upload
ios	flun.zip	20180712104449	upload
ios	mobile0712.ica	0	upload
macos	mobile0712.dmg	0	upload

	항목	설명
1	General	빌드할 앱 기본 정보를 입력합니다.
2	Build configuration	빌드 시 필요한 설정 옵션을 지정합니다.
3	Authority	"App" 사용 권한을 지정합니다.
4	Target OS	앱을 빌드할 대상을 선택합니다.
5	Nexacro application resource	앱에서 사용하는 리소스를 관리합니다.

## 5.2.1 General

**General**

- Package Name: [text field]
- Version: 1.0.0
- App Title: hello

	항목	설명
1	Package Name	Package 이름을 입력합니다.
2	Version	빌드할 앱 버전을 입력합니다.
3	App Title	빌드할 앱 이름을 입력합니다.



Package Name은 빌드한 앱을 구별하는 식별자입니다. 일반적으로 도메인 주소를 반대로 기재한 형태로 Package Name을 지정합니다.  
예) com.nexacro

Package Name을 하나의 단어로만 입력하는 경우에는 빌드 시 오류가 발생할 수 있습니다.



Target OS에 iOS가 포함된 경우에는 Provisioning Profile 발급 요청시 입력한 Package 명과 같은 값으로 Package Name을 입력하여야 합니다.

## 5.2.2 Build configuration

Build configuration	
* Nexacro Project URL	<input type="text" value="http://www.nexacro.com/2020/appbuilder/archives/283"/>
* Build mode	<input type="radio"/> debug <input checked="" type="radio"/> release
* Detail Error Message Output	<input type="radio"/> true <input checked="" type="radio"/> false
* Nexacro Mobile Library	<input type="text" value="lib_20180717_0"/> ▼

	항목	설명
1	Nexacro Project URL	넥사크로플랫폼 애플리케이션이 배포된 URL을 입력합니다. URL 입력 시 마지막 슬래시 표기는 생략합니다.
2	Build mode	빌드 모드를 선택합니다. "debug" 모드 선택 시 ADB(Android Debug Bridge)를 통한 디버깅 작업을 수행할 수 있습니다.
3	Detail Error Message Output	앱 로딩 중 발생하는 오류 메시지를 사용자에게 노출할지 설정
4	Nexacro Mobile Library	넥사크로플랫폼 모바일 라이브러리를 선택합니다. 애플리케이션 개발 시 사용한 넥사크로플랫폼과 같은 버전의 라이브러리를 선택해야 합니다.

## 5.2.3 Authority

Authority							
* Owner	<input type="text" value="admin"/>						
	<input type="radio"/> share <input type="radio"/> public <input checked="" type="radio"/> private <input type="button" value="-"/> <input type="button" value="+"/>						
* Access	<table border="1"> <thead> <tr> <th></th> <th>ID</th> <th>Authority</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td></td> <td></td> </tr> </tbody> </table>		ID	Authority	<input type="checkbox"/>		
	ID	Authority					
<input type="checkbox"/>							

	항목	설명
1	Owner	"App" 소유자를 지정합니다. (로그인 계정으로 자동 입력됩니다).
2	Access	"App" 사용 권한을 지정합니다. 사용 권한에 따라 [App List]에 표시되는 목록이 달라집니다. - share: 지정된 사용자만 사용할 수 있습니다. - public: 모든 사용자가 사용할 수 있습니다. - private: Owner만 사용할 수 있습니다.



Access 설정과 상관없이 Admin 권한을 가진 사용자는 모든 "App"을 사용할 수 있습니다.

## 5.2.4 Target OS

**Target OS**

<input checked="" type="checkbox"/> Android	<input type="text" value="start_android.json"/>	Upload	detail
<input type="checkbox"/> iOS	<input type="text"/>	Upload	detail
<input type="checkbox"/> macOS	<input type="text"/>	Upload	detail

앱을 빌드할 대상 운영체제를 선택합니다. 선택한 운영체제에 대한 세부 설정을 지정할 수 있습니다. 넥사크로 스튜디오가 아니라 관리 콘솔에서 직접 리소스를 업로드하는 경우에는 선택한 운영체제에 맞는 start\_(os\_name).json 파일을 등록하면 [Nexacro application resource] 항목에 필요한 리소스 목록을 출력하고 리소스 파일을 업로드할 수 있습니다.

[detail] 버튼을 클릭하면 json 파일 내용을 표 형태로 보여줍니다.

**Target OS**

<input checked="" type="checkbox"/> Android	<input type="text" value="start_android.json"/>	Upload	detail
---	---	--------	--------

**Nexacro Application Resource**

os	update_url	engine_url	engine_setupkey	engine_version	timeout	retry	auto_update	description
android					30	3	auto	
common	http://172.10.11.89:8080/appbul				30	3	auto	

**Nexacro Application Resource**

type	file	target_path	version	fail_pass	description
Theme	default.zip	./_resource/_t	0.0.0.0		
Engine	nexacro17lib.zip		0.0.0.0		
File	Archive03.xzip		0.0.0.0		

OK

## 5.2.5 Nexacro application resource

넥사크로 스튜디오가 아니라 관리 콘솔에서 직접 리소스를 업로드하는 경우에는 [Target OS] 항목에 선택한 운영체제에 맞는 start\_(os\_name).json 파일을 등록하면 필요한 리소스 목록을 출력하고 리소스 파일을 업로드할 수 있습니다. 넥사크로 스튜디오에서 App Builder로 빌드를 처리한 적이 있는 경우에는 넥사크로 스튜디오에서 등록한 리소스 파일 목록을 출력합니다.

새로 리소스 파일을 등록하거나 json 파일에 등록된 내용과 버전이 다른 경우에는 "changed data"로 색상을 구분해 표시합니다.

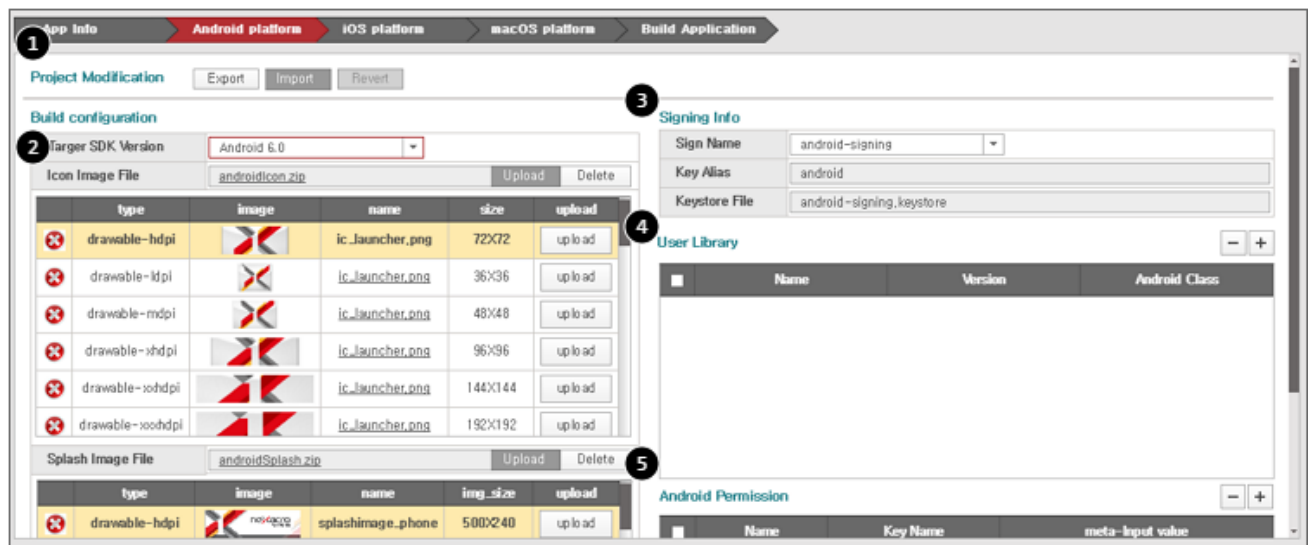
Target OS				
<input checked="" type="checkbox"/> Android	<a href="#">start_android.json</a>	Upload	detail	
<input checked="" type="checkbox"/> iOS	<a href="#">start_ios.json</a>	Upload	detail	
<input checked="" type="checkbox"/> macOS	<a href="#">start_macos.json</a>	Upload	detail	

Nexacro application resource				
				<span style="color: orange;">■</span> changed data
	os	name	version	upload
	android   ios	<a href="#">userfont00_xfont.xzip</a>	0.0.0.0	upload
	ALL	<a href="#">nexacro17lib.zip</a>	0.0.0.0	upload
	ios   android	<a href="#">ios.zip</a>	0.0.0.0	upload
	ALL	<a href="#">default.zip</a>	0.0.0.0	upload
	ALL	<a href="#">Archive00.xzip</a>	0.0.0.0	upload
	ios	Run.zip	0.0.0.0	upload

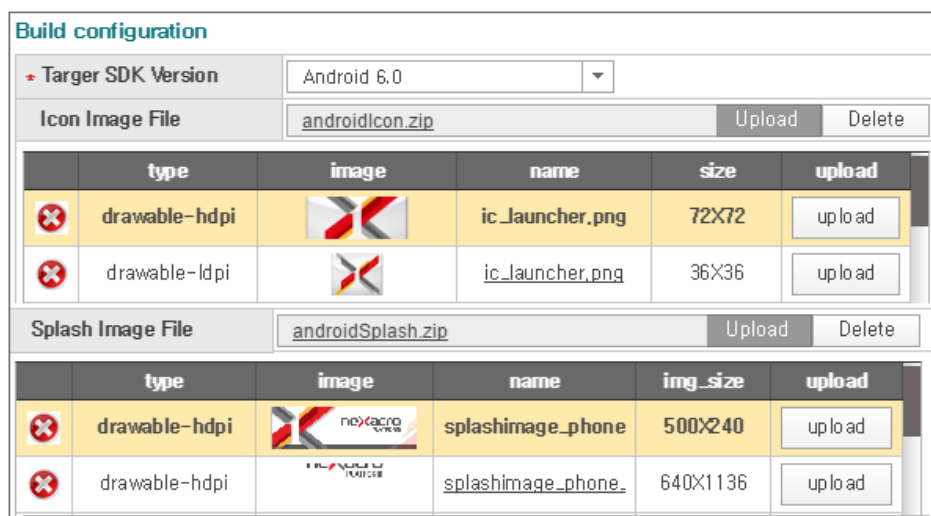


## 5.3 Android platform



	항목	설명
1	Project Modification	생성된 "App" 정보를 안드로이드 스튜디오에서 편집할 수 있는 프로젝트 형태로 내보내거나 안드로이드 스튜디오에서 편집한 내용을 가져올 수 있습니다.
2	Build configuration	Target SDK를 선택하고 Icon, Splash 이미지를 등록합니다.
3	Signing Info	Signing 정보를 선택합니다.
4	User Library	User Library를 선택합니다.
5	Android Permission	디바이스 API 권한을 선택합니다.

### 5.3.1 Build Configuration



	항목	설명
1	Target SDK Version	빌드할 앱의 Android API Level을 선택합니다. 최소버전은 Android 4.4.2(API Level 19)로 고정되어 있습니다.
2	Icon Image File	앱 아이콘 이미지 파일을 등록합니다. 별도로 파일을 지정하지 않으면 넥사크로플랫폼 기본 이미지가 적용됩니다. 여러 이미지 파일을 압축한 후 하나의 압축파일(zip 파일)로 등록하거나 개별 파일을 등록할 수 있습니다.
3	Splash Image File	앱 실행 시 표시되는 Splash 이미지 파일을 등록합니다. 별도로 파일을 지정하지 않으면 넥사크로플랫폼 기본 이미지가 적용됩니다. 여러 이미지 파일을 압축한 후 하나의 압축파일(zip 파일)로 등록하거나 개별 파일을 등록할 수 있습니다.



압축파일로 등록하는 경우에는 압축파일 내 폴더와 파일명을 아래와 같이 작성해야 합니다.

ic\_launcher.png: 아이콘 이미지 파일

splashimage\_phone\_landscape.png: 가로 방향 Splash 이미지 파일

splashimage\_phone\_portrait.png: 세로 방향 Splash 이미지 파일



폴더명과 파일명이 틀릴 경우 정상 빌드가 되지 않거나 앱 실행 중 정상적으로 동작하지 않을 수 있습니다.

Directory	file name	Image Size(px)
drawable-hdpi	ic_launcher.png	72x72
	splashimage_phone_landscape.png	화면에 맞게
	splashimage_phone_portrait.png	
drawable-ldpi	ic_launcher.png	36x36
	splashimage_phone_landscape.png	
	splashimage_phone_portrait.png	
drawable-mdpi	ic_launcher.png	48x48
	splashimage_phone_landscape.png	
	splashimage_phone_portrait.png	
drawable-xhdpi	ic_launcher.png	96x96
	splashimage_phone_landscape.png	
	splashimage_phone_portrait.png	
drawable-xxhdpi	ic_launcher.png	144x144
	splashimage_phone_landscape.png	
	splashimage_phone_portrait.png	
drawable-xxxhdpi	ic_launcher.png	192x192
	splashimage_phone_landscape.png	
	splashimage_phone_portrait.png	

### 5.3.2 Signing Info

Signing Info	
Sign Name	android-signing ▼
Key Alias	android
Keystore File	android-signing.keystore

메뉴[Signing]에서 등록한 Singning 항목 중 앱 빌드에 사용할 항목을 선택합니다.

### 5.3.3 User Library

메뉴[User Library]에서 등록한 User Library 항목 중 앱 빌드에 사용할 항목을 선택합니다.

### 5.3.4 Android Permission

넥사크로플랫폼 디바이스 API 권한을 선택합니다. [+] 버튼을 클릭하면 전체 디바이스 API 목록이 표시되며 필요한 권한을 선택한 후 [OK] 버튼을 클릭합니다.

Device Api Permission List		
<input type="checkbox"/>	Name	permission
<input type="checkbox"/>	Contact	android.permission.READ_CONTACTS android.permission.WRITE_CONTACTS
<input checked="" type="checkbox"/>	Map	com.google.android.providers.gsf.permission.READ_GSERVICES
<input type="checkbox"/>	SMS	android.permission.SEND_SMS android.permission.RECEIVE_SMS android.permission.READ_SMS android.permission.WRITE_SMS
<input type="checkbox"/>	ExternalAPI	android.permission.GET_TASKS

Ok Cancel

Map, X-PUSH를 사용하는 경우에는 추가로 키값을 입력해주어야 합니다.

Android Permission			
<input type="checkbox"/>	Name	Key Name	meta-Input value
<input type="checkbox"/>	Camera		
<input type="checkbox"/>	Geolocation		
<input type="checkbox"/>	Map	API KEY	<input type="text"/>

## 5.4 iOS platform

The screenshot shows the Xcode project configuration interface for the iOS platform. The interface is divided into four numbered sections:

- Project Modification**: Includes tabs for App Info, Export, Import, and Revert.
- Build configuration**: Includes fields for Minimum SDK (6.0), Icon Image File (iosRedIcon.zip), and Splash Image File (splashimageRed.zip). Below these are tables for icon and splash image uploads.
- Signing Info**: Includes fields for Sign Name (ios-inhouse), Certificate Name (iPhone Distribution: TOBESOFT), Certificate File (TOBESOFT\_In-House\_Distribution\_ServerTeam.p12), and Provisioning Profile (TOBESOFT\_InHouse\_Distribution\_ServerTeam.mobileprovision).
- User Library**: Includes a table for selecting a user library.

	항목	설명
1	Project Modification	생성된 "App" 정보를 Xcode에서 편집할 수 있는 프로젝트 형태로 보내거나 Xcode에서 편집한 내용을 가져올 수 있습니다.
2	Build configuration	Target SDK를 선택하고 Icon, Splash 이미지를 등록합니다.
3	Signing Info	Signing 정보를 선택합니다.
4	User Library	User Library를 선택합니다.

## 5.4.1 Build configuration

### Build configuration

★ Minimum SDK

6.0

Icon Image File

iosRedIcon.zip

Upload

Delete

	image	name	size	upload
		Icon.png	57X57	<div>upload</div>
		Icon@2x.png	114X114	<div>upload</div>

Splash Image File

splashimageRed.zip

Upload

Delete

	image	name	img_size	upload
		splashimage_pad_landscape.	1024X748	<div>upload</div>
		splashimage_pad_portrait.png	768X1004	<div>upload</div>

	항목	설명
1	Minimum SDK	
2	Icon Image File	앱 아이콘 이미지 파일을 등록합니다. 별도로 파일을 지정하지 않으면 넥사크로플랫폼 기본 이미지가 적용됩니다. 여러 이미지 파일을 압축한 후 하나의 압축파일(zip 파일)로 등록하거나 개별 파일을 등록할 수 있습니다.
3	Splash Image File	앱 실행 시 표시되는 Splash 이미지 파일을 등록합니다. 별도로 파일을 지정하지 않으면 넥사크로플랫폼 기본 이미지가 적용됩니다. 여러 이미지 파일을 압축한 후 하나의 압축파일(zip 파일)로 등록하거나 개별 파일을 등록할 수 있습니다.



아이콘, Splash 파일명은 아래와 같이 작성해야 합니다.

file name	Image Size(px)	설명
Icon.png	57x57	Home screen on iPhone/iPod touch (iOS 6.1 and earlier)
Icon@2x.png	114x114	Home screen on iPhone/iPod Touch with retina display (iOS 6.1 and earlier)
splashimage_pad_landscape.png	1024x768	Pad 가로 방향 Splash 이미지 파일
splashimage_pad_portrait.png	768x1024	Pad 세로 방향 Splash 이미지 파일
splashimage_phone_landscape	960x640	Phone 가로 방향 Splash 이미지 파일

file name	Image Size(px)	설명
pe.png		
splashimage_phone_portrait.png	640x960	Phone 세로 방향 Splash 이미지 파일
splashimgae_phone5_landscape.png	1136x640	Phone 가로 방향 Splash 이미지 파일
splashimgae_phone5_portrait.png	640x1136	Phone 세로 방향 Splash 이미지 파일

## 5.4.2 Signing Info

**Signing Info**

<b>Sign Name</b>	ios-inhouse ▼
<b>Certificate Name</b>	iPhone Distribution: [redacted]
<b>Certificate File</b>	[redacted].In-House_Distribution_ServerTeam.p12
<b>Provisioning Profile</b>	[redacted].InHouse_Distribution_ServerTeam.mobileprovision

메뉴[Signing]에서 등록한 Singning 항목 중 앱 빌드에 사용할 항목을 선택합니다.

## 5.4.3 User Library

메뉴[User Library]에서 등록한 User Library 항목 중 앱 빌드에 사용할 항목을 선택합니다.

## 5.5 macOS platform

The screenshot shows the Mobile App Builder interface for the macOS platform. The interface is divided into three main sections: 1. Project Modification, 2. Build configuration, and 3. Signing Info. Section 1 includes buttons for Export, Import, and Revert. Section 2 shows the Minimum SDK set to 6.0, and fields for Icon Image File (.icns), Splash Image File (.png), and Dmg Image File (.png). Section 3 shows the Developer ID selected, with fields for Sign Name, Certificate Name, and Certificate File.

	항목	설명
1	Project Modification	생성된 "App" 정보를 Xcode에서 편집할 수 있는 프로젝트 형태로 내보내거나 Xcode에서 편집한 내용을 가져올 수 있습니다.
2	Build configuration	Minimum SDK를 선택하고 Icon, Splash, Dmg 이미지를 등록합니다.
3	Signing Info	Signing 정보를 선택합니다.

### 5.5.1 Build configuration

#### Build configuration

The screenshot shows the Build configuration section of the Mobile App Builder interface. It includes the following fields and options:

- Minimum SDK:** 6.0
- Icon Image File (.icns):** icon.icns (with Upload and Delete buttons)
- Splash Image File (.png):** nexacro\_splash.png (with Upload and Delete buttons)
- Dmg Image File (.png):** install-mac.png (with Upload and Delete buttons)

The Dmg Image File section also displays a preview of the Mobile App Builder logo and a note: "Drag to install APP in your Applications folder."

	항목	설명
1	Minumum SDK	
2	Icon Image File	앱 아이콘 이미지 파일을 등록합니다. 별도로 파일을 지정하지 않으면 넥사크로플랫폼 기본 이미지가 적용됩니다. 파일 확장자는 .icns 입니다.
3	Splash Image File	앱 실행 시 표시되는 Splash 이미지 파일을 등록합니다. 별도로 파일을 지정하지 않으면 넥사크로플랫폼 기본 이미지가 적용됩니다.
4	Dmg Image File	앱 설치 시 표시되는 이미지 파일을 등록합니다. 별도로 파일을 지정하지 않으면 넥사크로플랫폼 기본 이미지가 적용됩니다.



Dmg Image File에 등록한 이미지 파일의 DPI가 72X72가 아닌 경우 Mac OS 10.7 이상에서 배경이 왜곡되어 보여질 수 있습니다.

## 5.5.2 Signing Info

**Signing Info**

☒ Developer ID
 ☐ none

<b>Sign Name</b>	osx-signing
<b>Certificate Name</b>	Developer ID Application: [redacted]
<b>Certificate File</b>	developer-id-[redacted].p12

메뉴[Signing]에서 등록한 Singning 항목 중 앱 빌드에 사용할 항목을 선택합니다.



"none" 항목을 선택한 경우 Signing Info를 포함하지 않고 앱 빌드를 처리합니다. 빌드는 정상적으로 동작하지만 앱 설치 후 실행 시 보안 관련 설정을 변경해주어야 합니다.

[https://support.apple.com/kb/PH14369?locale=en\\_US](https://support.apple.com/kb/PH14369?locale=en_US)

## 5.6 Project Modification

App Builder에서 지원하지 못하는 기능을 필요로 하는 경우 생성된 "App" 정보를 프로젝트 형태로 내보내 각 운영 체제를 지원하는 개발도구에서 필요한 기능을 추가하거나 편집할 수 있습니다. [Project Modification] 기능을 사용하면 현재 "App" 정보를 압축 파일 형태로 내보내거나 외부 개발도구에서 수정한 프로젝트를 가져올 수 있습니다.





	항목	설명
1	Export	현재 "App" 프로젝트를 압축파일로 내보냅니다.
2	Import	현재 "App" 프로젝트에 사용자가 수정한 프로젝트를 가져옵니다.
3	Revert	App Builder에 저장된 데이터로 원복합니다.

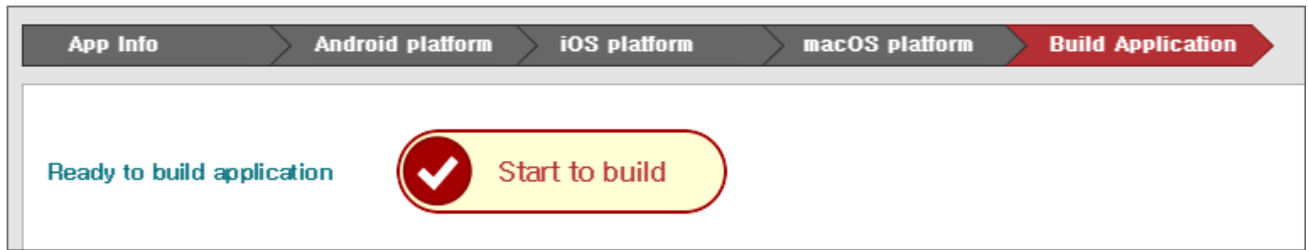
[Import] 버튼 클릭 시에는 두 가지 옵션 중 하나를 선택할 수 있습니다. 버튼 클릭 시 아래 그림처럼 옵션을 선택할 수 있는 창이 표시되며 [YES], [NO] 버튼 선택에 따라 수정한 프로젝트를 가져오는 방식을 결정합니다.



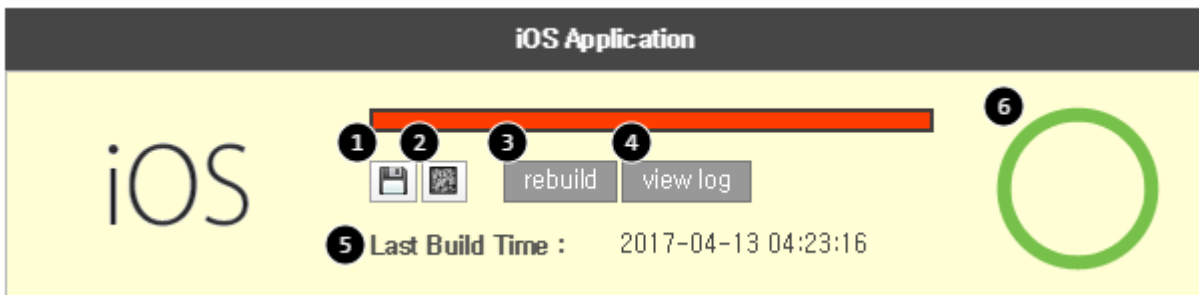
	항목	설명
1	YES	사용자가 수정한 후 Import 한 프로젝트의 설정을 유지합니다. - 프로젝트의 각 플랫폼 별 설정화면의 상태를 비활성화 합니다. - 공통 설정 항목은 수정 가능하나 해당 플랫폼은 적용하지 않습니다.
2	NO	사용자가 수정한 후 Import 한 프로젝트의 설정을 유지 하지 않습니다. - 프로젝트의 각 플랫폼 별 설정 화면의 상태를 활성화 합니다. - Import 된 프로젝트에 AppBuilder 콘솔화면에 명시된 설정을 덮어쓰고 빌드합니다.

## 5.7 Build App

새로 "App"을 생성한 경우에는 [Start to build]라는 버튼이 표시됩니다. 버튼을 클릭하면 선택한 운영체제에 해당하는 앱 빌드를 시작합니다.



빌드 결과를 표시하며 빌드를 성공한 경우에는 생성된 앱을 내려받을 수 있는 링크를 생성합니다. [Rebuild All] 버튼을 클릭해 전체 앱을 다시 빌드하거나 운영체제마다 따로 빌드할 수 있는 기능도 지원합니다.



	항목	설명
1	다운로드	생성된 앱 설치 파일을 내려받습니다.
2	QRCode	생성된 앱 설치 파일을 내려받을 수 있는 URL, QRCode를 표시합니다.
3	rebuild	선택된 운영체제 앱을 다시 빌드합니다.
4	view log	앱 빌드 시 생성된 로그를 확인할 수 있습니다.
5	Last Build Time	마지막으로 앱을 빌드한 시간을 표시합니다.
6	빌드 여부	빌드 성공 또는 실패 여부를 표시합니다. "O"로 표시된 경우에는 빌드 성공, "X"로 표기된 경우에는 빌드 실패입니다.



관리 콘솔에서 값이 변경된 경우에는 앱 빌드 전에 [Save] 버튼을 클릭해 변경된 내용을 저장해야 합니다. 저장되지 않은 정보는 빌드 시 반영되지 않습니다.



설치 파일명은 아래와 같이 생성됩니다. "App Title", "Build mode" 정보는 [App Info] 항목에서 설정합니다.

Android: [App Title]-[Build mode].apk

iOS, macOS: [App Title].ipa

예) sample-debug.apk, sample.ipa

모바일 디바이스에는 "App Title" 정보에 지정한 이름으로 설치됩니다.



QRCode 기능을 제공하는 앱을 사용하는 경우 QRCode를 인식한 후 URL을 Chrome 브라우저에서 처리하는 과정에서 "downloadFile.do" 또는 "downloadFile.htm"으로 주소창에 표시되고 파일을 내려받지 못하는 경우가 있습니다. 이런 경우에는 아래와 같이 조치합니다.

- Chrome 브라우저 캐시 삭제
- Chrome 브라우저를 최신 버전으로 업데이트

## 6.

# App 목록

생성된 "App" 목록을 확인할 수 있습니다. 관리 콘솔에서 메뉴[App > App List] 항목을 선택합니다. [App List]에서는 로그인한 사용자가 권한을 가지고 있는 "App" 목록만 표시됩니다.

App Name

Q Search

Project List

CopyDelete

	App Name	Package Name	Version	Device Type	Build	Download	Last Build Time	Owner	Access	
<input type="checkbox"/>	Project2	com.example.project2	1.0.0	Android	X			김민준	private	
				iOS	X					
				macOS	X					
<input type="checkbox"/>	Project2	com.example.project2	1.0.0	Android	O			2017-01-24 05:08:39	김민준	private
				iOS	O			2017-01-24 05:08:16		
				macOS	O			2017-01-24 05:08:57		

항목별로 빌드 상태와 빌드에 성공한 경우 설치 파일을 내려받을 수 있습니다. 항목을 선택한 후 오른쪽 상단 [Copy] 버튼을 클릭하면 "[App\_Name]\_copy"라는 이름으로 복사된 "App"을 생성합니다.



[Copy] 버튼을 클릭해 정보를 복사하는 경우 App Name은 수정할 수 없습니다.

## 파트 V.

---

### iOS/Android/macOS App

## 7.

---

# 앱 개발 및 실행 (iOS)

## 7.1 앱 개발 환경 설정

iOS용 앱은 Xcode, iOS SDK를 이용해 개발합니다. Xcode, iOS SDK는 Windows 계열의 PC에서는 사용할 수 없으며 Mac에서만 구동합니다. 그리고 Xcode를 내려받고 앱을 만들고 테스트를 하기 위해서는 Apple 개발자 계정이 필요합니다.

iOS 앱 개발 환경은 기본적으로 다음과 같은 절차에 의해 구성되며 세부 내용은 개발자 사이트 업데이트에 따라 변경될 수 있습니다.

1. 개발자 계정 등록
2. Xcode, iOS SDK 설치
3. 배포 시 필요한 작업

### 7.1.1 개발자 계정 등록

Apple 계정만 있으면 개발자 라이선스를 따로 발급받지 않아도 계정 등록만으로 Xcode를 내려받거나 기타 무료로 제공되는 서비스를 사용할 수 있습니다. 그리고 iOS에서는 시뮬레이터를 사용해 테스트를 진행할 수 있습니다.

시뮬레이터를 사용하는 경우에는 별도의 인증 절차를 거치지 않지만, 단말기를 연결하는 경우에는 추가적인 인증 절차가 필요하므로 개발자 라이선스를 발급받아야 합니다.

### Apple 계정 생성

개발자 계정을 등록하기 위해서는 먼저 Apple 계정이 있어야 합니다. 개발자 계정을 따로 만드는 것이 아니라 이미 만들어진 Apple 계정을 개발자 계정으로 등록하는 것입니다.

새로운 Apple 계정은 아래 링크에서 만들 수 있으며 등록 시 사용하는 이메일 계정을 ID로 사용합니다. 기존에 사용

하고 있는 Apple 계정이 있다면 이 단계는 지나갑니다.



<https://developer.apple.com/account/>

## 개발자 라이선스 인증

단말기 연결 없이 시뮬레이션으로만 테스트를 진행하고자 한다면 아래 링크에서 Apple 계정을 개발자 계정으로 등록할 수 있습니다.



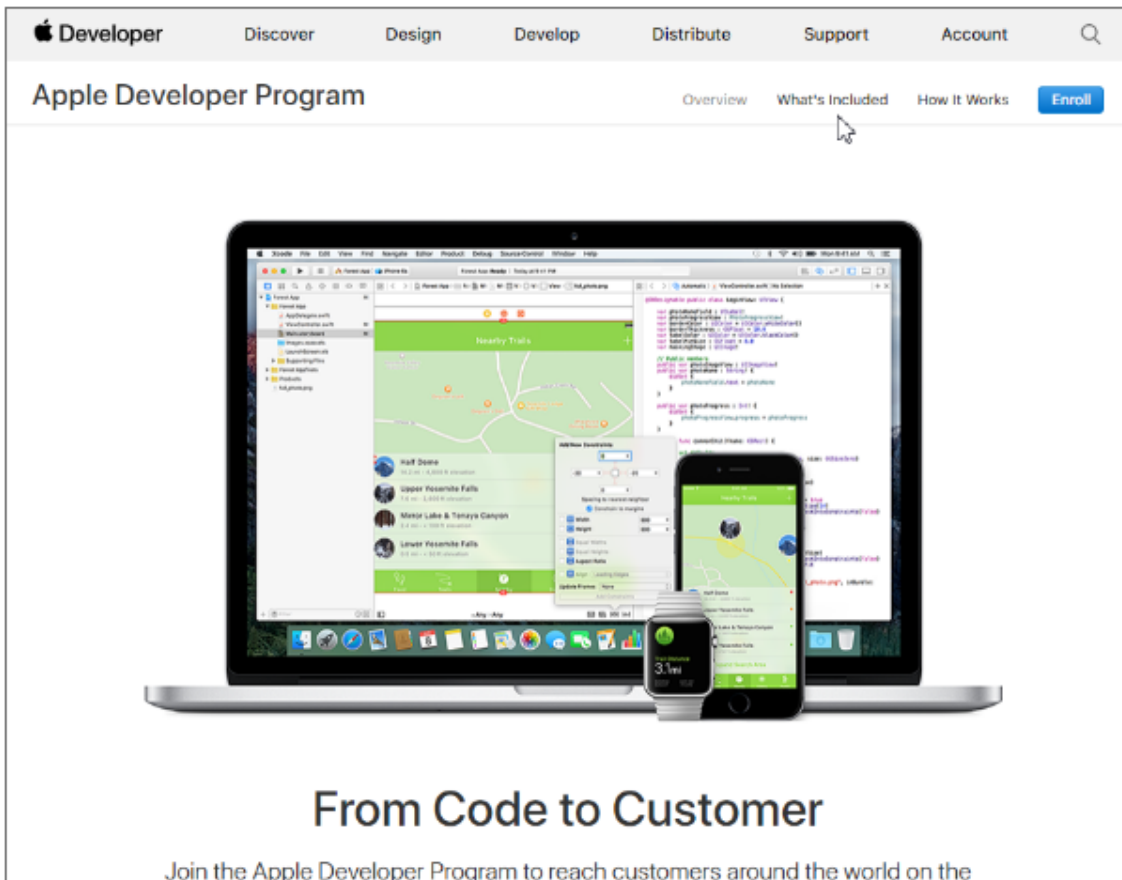
<https://developer.apple.com/register>

개발, 테스트, 배포에 필요한 서비스를 받으려면 iOS 개발자 프로그램에 가입해야 합니다. 1년 단위로 등록할 수 있으며 별도의 비용이 발생합니다. 제공되는 서비스에 대해서는 아래 링크를 참고하세요.



<https://developer.apple.com/programs/>

화면 상단에 있는 Enroll 버튼을 클릭하면 등록하기 과정이 시작됩니다. 개발자 계정은 개인으로 등록하거나 회사 명의로 등록할 수 있습니다.



회사 명의로 등록하는 경우에는 DUNS 번호가 필요합니다. DUNS 번호는 전 세계 기업을 관리하는 일종의 코드입니다. 애플 사이트에서 DUNS 번호를 별도 비용 없이 발급받을 수 있습니다. 사이트에서 신청 후 발급 대행사에서 유선으로 필요한 서류를 요청하고 해당 서류까지 전달해야 발급 처리가 완료됩니다.

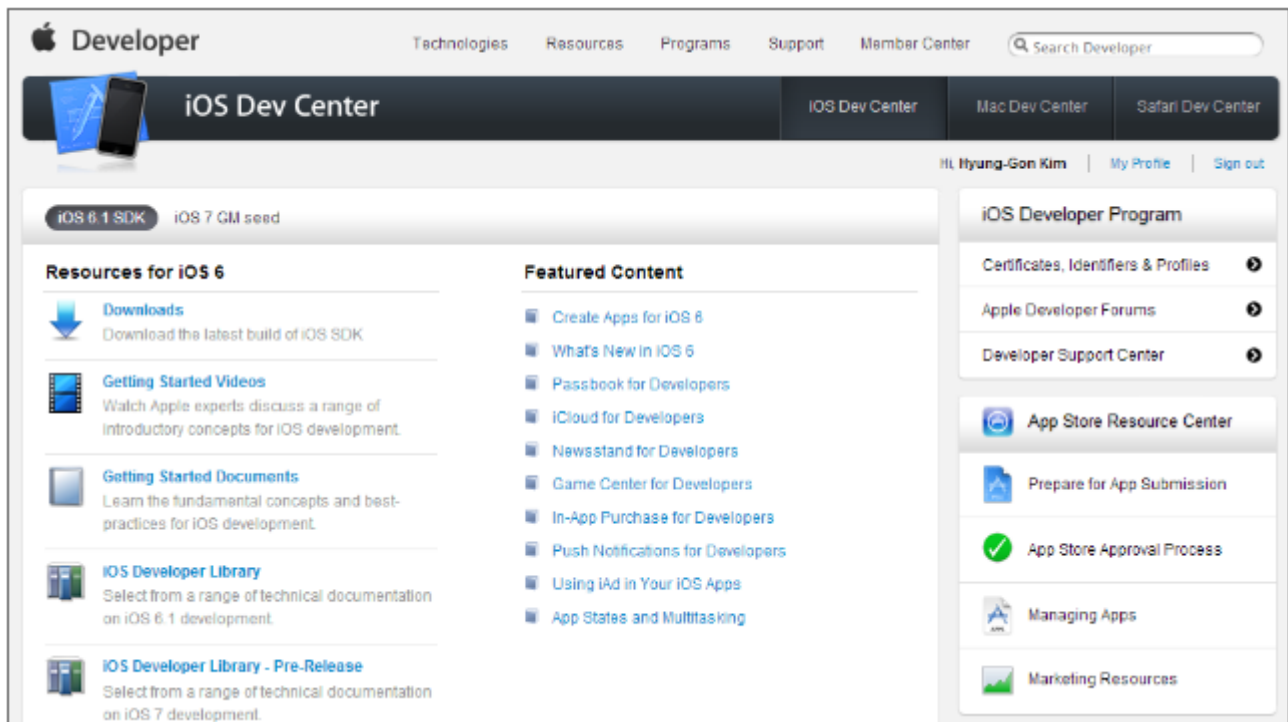


<https://developer.apple.com/ios/enroll/dunsLookupForm.action>  
<https://developer.apple.com/support/D-U-N-S/>

계정이 등록되면 Apple 담당자가 신청된 내용을 검토하고 유선으로 확인합니다. 다른 문제가 없다면 등록 완료 메일에 포함된 링크로 이동해 라이선스에 동의하고 스토어에서 1년짜리 개발자 계정을 구매할 수 있습니다.

구매 단계까지 마치면 이메일로 활성화 코드(Activation code)가 전달되고 인증 페이지에서 해당 코드를 등록하면 모든 절차가 완료됩니다.

iOS Dev Center에서 화면 오른쪽에 iOS Developer Program 메뉴가 추가된 것을 확인할 수 있습니다.



## 7.1.2 Xcode, iOS SDK 설치

Xcode는 맥에서 사용하는 통합개발환경으로 여러 가지 도구가 포함되어 있습니다. 이미 Xcode가 설치되어 있더라도 연결하려는 장비의 iOS SDK 버전에 따라 지원되는 Xcode 버전이 달라질 수 있으며 필요한 경우 업데이트를 해야 합니다.

내려받을 수 있는 최신 Xcode 버전은 아래 링크에서 확인할 수 있습니다. 링크를 통하지 않고 직접 앱스토어에서 내려받을 수도 있습니다. Xcode 설치 파일에는 iOS SDK와 Mac SDK가 포함되어 있습니다.





<https://developer.apple.com/xcode/>  
<https://developer.apple.com/download/>

## Downloads

Get the latest beta releases of Xcode, macOS, iOS, watchOS, tvOS, and more.

Release Software	Build	Date
<b>Xcode 9.2</b> <small>Includes macOS, iOS, watchOS, and tvOS SDKs.</small>	<a href="#">Release Notes</a> 9C40b	Dec 4, 2017 <a href="#">Download</a>

만일 다른 버전의 Xcode가 필요하다면 아래 링크에서 내려받을 수 있습니다.



<https://developer.apple.com/download/more/>

### 7.1.3 배포에 필요한 작업

실제 단말기와 연결해 테스트를 진행하기 위해서는 몇 가지 인증 절차가 필요합니다. 이와 관련된 작업을 위해서는 개발자 프로그램에 등록되어 있어야 합니다.

필요한 작업과 관련된 내용은 아래 링크를 참고하세요.



<https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Introduction/Introduction.html>

## 7.2 앱 프로젝트 개발

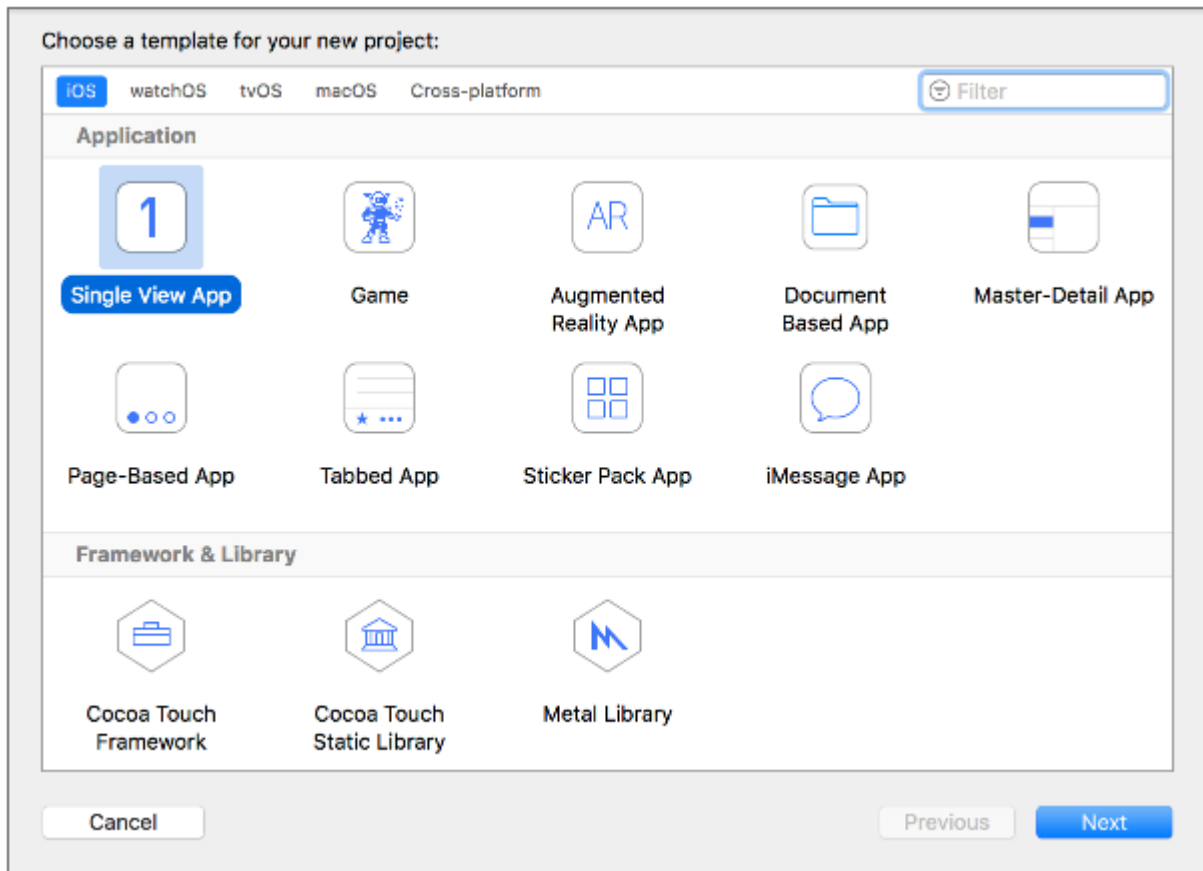
Xcode에서 아래와 같은 절차에 따라 iOS용 앱 프로젝트를 개발합니다. 앱 프로젝트를 진행하기 전에 넥사크로 스튜디오에서 개발된 애플리케이션에서 만들어진 아카이브 파일은 지정된 경로에 위치해야 합니다.

## 7.2.1 프로젝트 생성

넥사크로플랫폼으로 개발된 애플리케이션을 담을 iOS 프로젝트를 생성하고 기본 환경을 설정해야 합니다. 새로운 프로젝트는 아래 메뉴에서 생성할 수 있습니다.

File > New > Project

프로젝트 생성을 위한 템플릿 화면에서 'Single View App' 항목을 선택합니다.



Product Name과 필요한 항목을 지정한 후 Next 버튼을 클릭합니다. 열린 팝업 창에서 프로젝트를 생성할 위치를 지정한 후 'Create' 버튼을 클릭하면 새로운 프로젝트가 생성됩니다.

Choose options for your new project:

Product Name: HelloiOS

Team: TOBESOFT

Organization Name: MobileTeam

Organization Identifier: com.nexacro

Bundle Identifier: com.nexacro.HelloiOS

Language: Objective-C

☐ Use Core Data

☐ Include Unit Tests

☐ Include UI Tests

Cancel Previous Next



설치된 Xcode 버전에 따라 프로젝트 생성 방법이 조금씩 달라질 수 있습니다.



iPhone X 디바이스를 사용하는 경우에 위쪽과 아래쪽에 여백이 생기는 현상을 방지하기 위해서는 프로젝트에 LaunchScreen 스토리 보드를 추가해야 합니다.

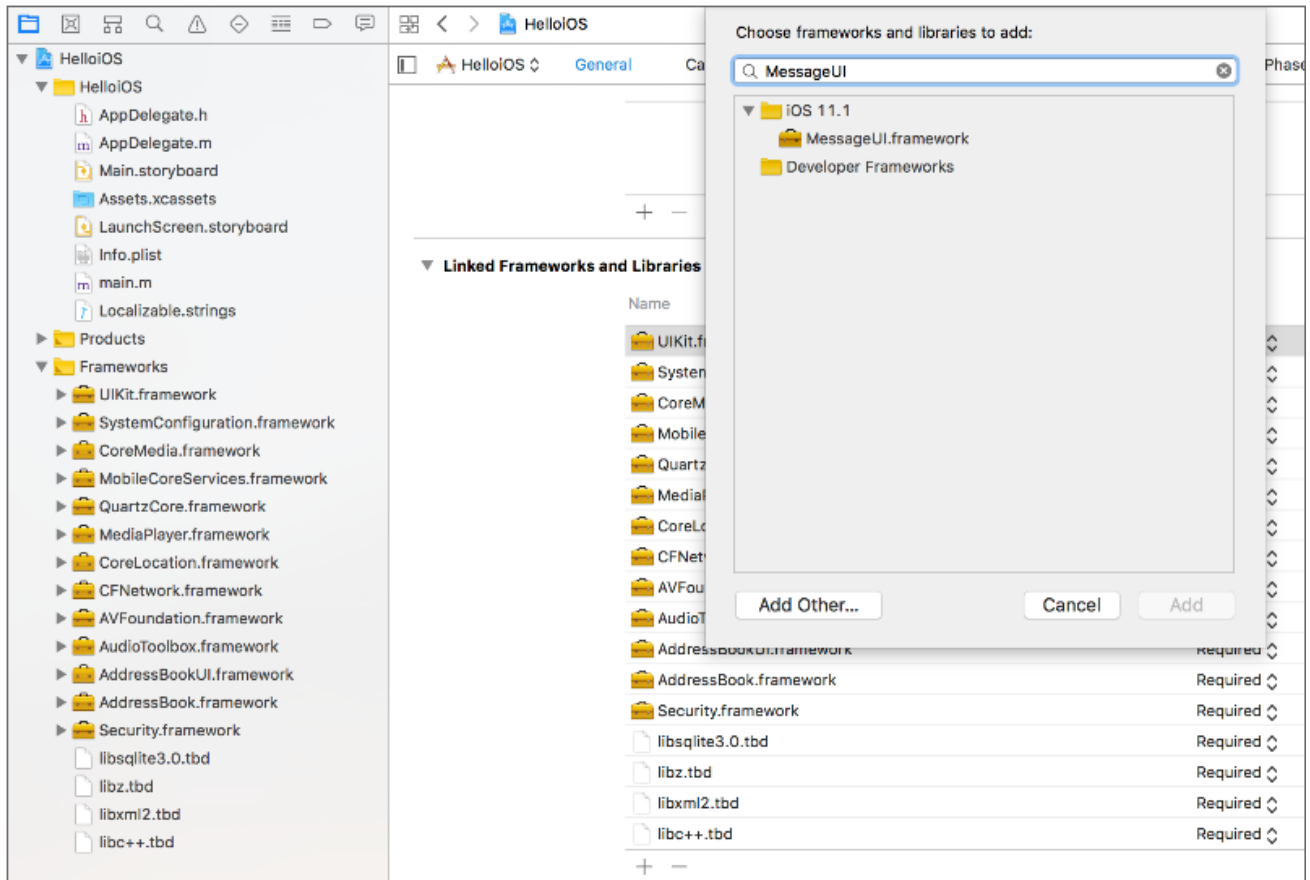
자세한 내용은 아래 링크를 참고해주세요.

<https://developer.apple.com/ios/update-apps-for-iphone-x/>

<https://developer.apple.com/ios/human-interface-guidelines/icons-and-images/launch-screen/>

## 7.2.2 iOS 라이브러리 및 프레임워크 설정

iOS에서 제공하는 API를 사용하는 데 필요한 라이브러리와 프레임워크를 설정합니다. 프로젝트를 선택하고 TARGETS 항목을 선택하면 화면 상단에 'General'이라는 탭을 확인할 수 있습니다. 해당 탭을 선택하면 'Linked Frameworks and Libraries' 항목이 보이는데 이 곳에서 라이브러리와 프레임워크를 추가합니다.



하단에 있는 + 버튼을 클릭해 아래와 같이 19개 항목을 추가합니다.



AddressBookUI.framework  
 AddressBook.framework  
 AudioToolbox.framework  
 AVFoundation.framework  
 CFNetwork.framework  
 CoreLocation.framework  
 MediaPlayer.framework  
 QuartzCore.framework  
 MobileCoreServices.framework  
 CoreMedia.framework  
 Security.framework  
 SystemConfiguration.framework  
 MessageUI.framework  
 CoreBluetooth.framework  
 UIKit.framework  
 libc++.tbd  
 libxml2.tbd  
 libz.1.2.5.tbd

libsqlite3.0.tbd

최초로 라이브러리나 프레임워크를 추가하면 Xcode의 프로젝트 네비게이터에 자동으로 Frameworks 그룹이 생성됩니다. 논리적으로 구성된 그룹이므로 실제 프로젝트 폴더에서는 보이지 않습니다.

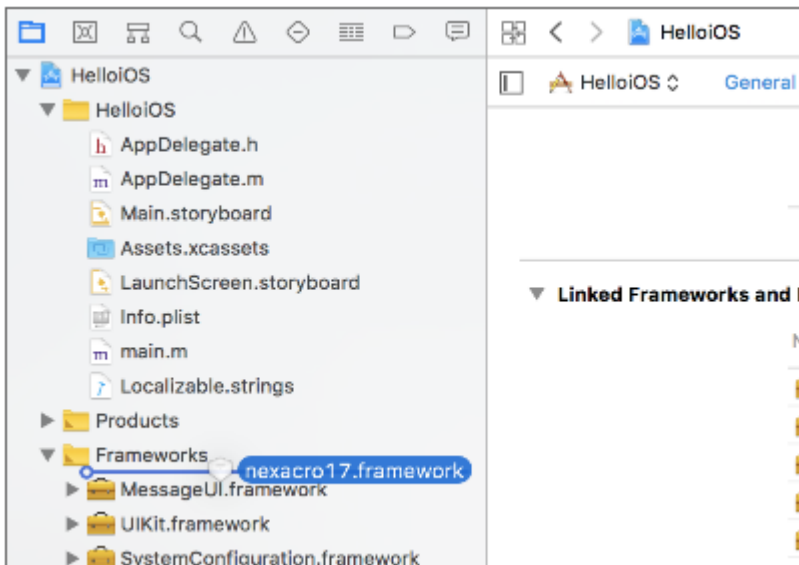


간혹 추가한 라이브러리와 프레임워크가 왼쪽의 프로젝트 네비게이터에만 보이고 'Linked Frameworks and Libraries'에는 안 보이는 경우가 있는데 이 때는 누락된 라이브러리와 프레임워크를 Project Navigator에서 'Linked Frameworks and Libraries'로 드래그 앤 드롭으로 추가해 주십시오. 그렇지 않으면 빌드시 에러가 발생합니다.

## 7.2.3 넥사크로플랫폼 라이브러리 설정

iOS 프로젝트에서 넥사크로플랫폼에 최적화된 환경을 만들기 위해 추가로 제공되는 넥사크로플랫폼 라이브러리 파일을 설정합니다.

넥사크로플랫폼 라이브러리는 압축 파일 형태로 제공되며 nexacro17.ios.framework.zip이라는 파일명으로 제공됩니다. 제공되는 파일은 압축을 풀어 생성된 프로젝트의 Frameworks 그룹으로 끌어다 놓습니다.



nexacro17.framework 파일은 2가지 형태로 제공되며 디바이스나 시뮬레이터에 따라 다른 파일을 지정해주어야 합니다. 시뮬레이터를 사용하는 경우에는 i386 파일을 사용하고 iPhone 3GS 이상의 디바이스에 연결할 때는 armv7 파일을 사용합니다. 파일 설정은 제품 버전에 따라 달라질 수 있습니다.

## 7.2.4 리소스 설정

앱에서 사용할 로딩 이미지, 아이콘, 메시지, 레이아웃 등을 설정하는 단계입니다. 진행하는 프로젝트에 따라 변경해 적용할 수 있습니다.

### 이미지 설정

Xcode의 템플릿으로 생성한 프로젝트에는 리소스를 관리하는데 필요한 빈 애셋 카탈로그(Asset Catalog)가 포함됩니다. 사용자는 필요에 따라 앱 아이콘, 런처 이미지 등을 애셋 카탈로그에 등록하여 사용할 수 있습니다.

iOS 앱에서 사용하는 앱 아이콘과 런처 스크린 이미지는 개별적으로 설정할 수 있으며 미리 만들어진 이미지를 연결할 수 있습니다.

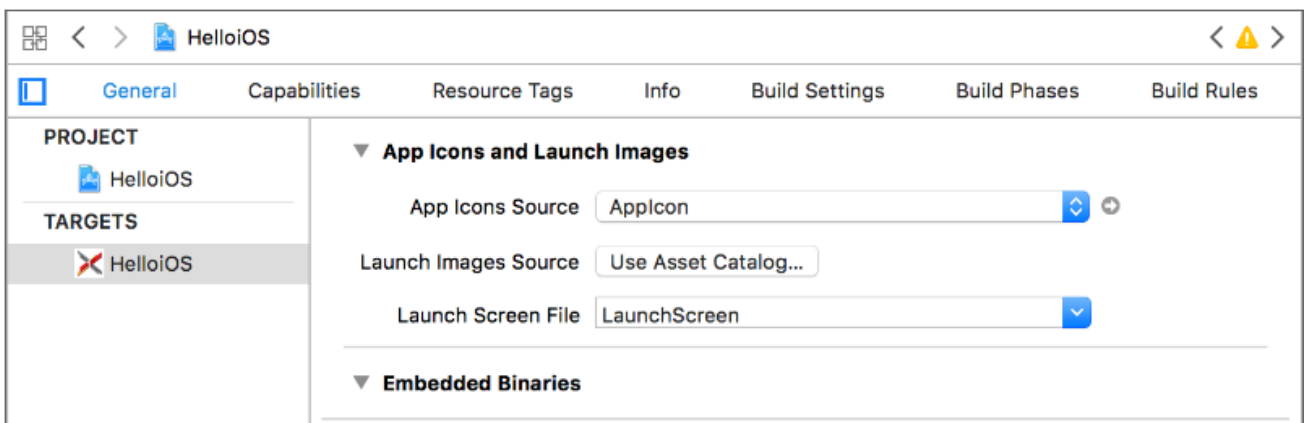


이미지 설정시 반드시 해당 iOS 단말기에 맞게 제작된 이미지를 사용해야 합니다. 단말기에 맞지 않는 크기의 이미지 혹은 사양과 맞지 않는 속성을 갖는 이미지를 설정하면 앱 빌드시 에러가 발생하거나 정상적으로 동작하지 않습니다.

아이콘 및 이미지 설정과 관련된 자세한 내용은 애플 개발자 사이트를 참고하세요.

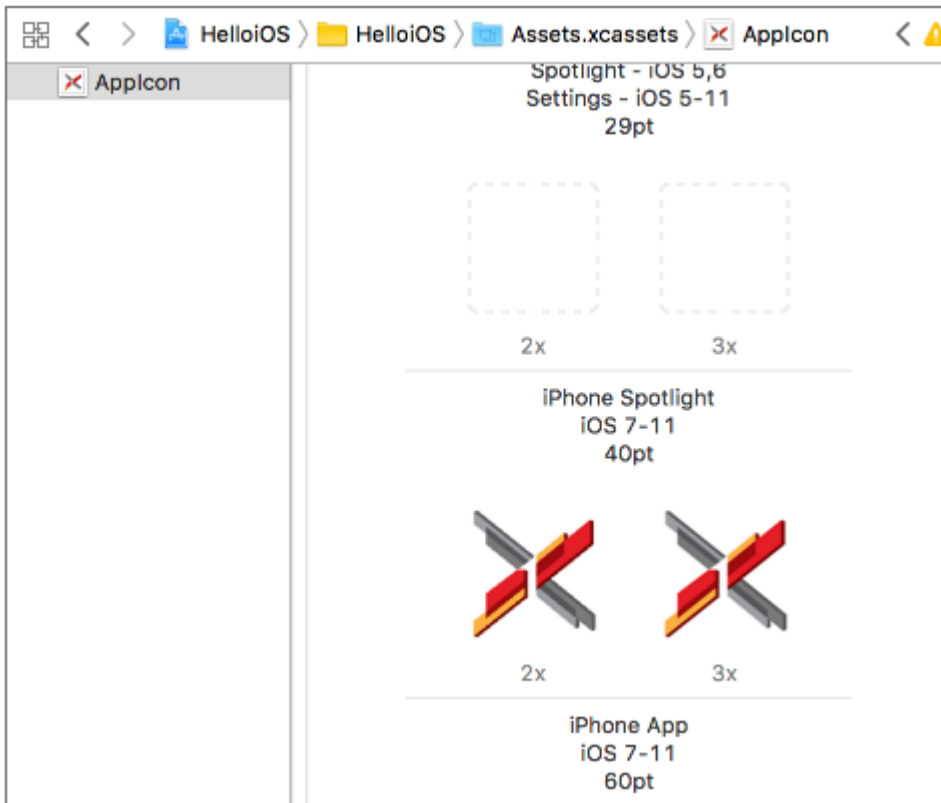
<https://developer.apple.com/ios/human-interface-guidelines/icons-and-images/image-size-and-resolution/>

아이콘 및 이미지 설정은 프로젝트 설정에서 할 수 있습니다. 프로젝트를 선택하고 TARGETS 항목을 선택하면 화면 상단에 'General'이라는 탭을 확인할 수 있습니다. 해당 탭을 선택하고 'App Icons and Launch Images' 항목을 확인합니다.



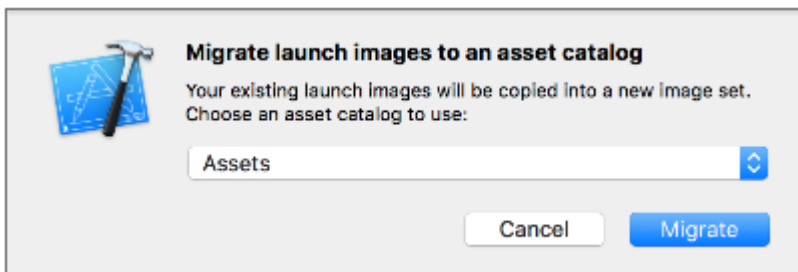
- 앱 아이콘 설정

'App Icons Source' 항목 옆의 화살표를 클릭하면 다음과 같이 애셋 카탈로그에서 앱 아이콘을 설정할 수 있습니다. 미리 제작한 아이콘 이미지를 단말기와 용도에 맞는 위치에 드래그 앤 드롭합니다.

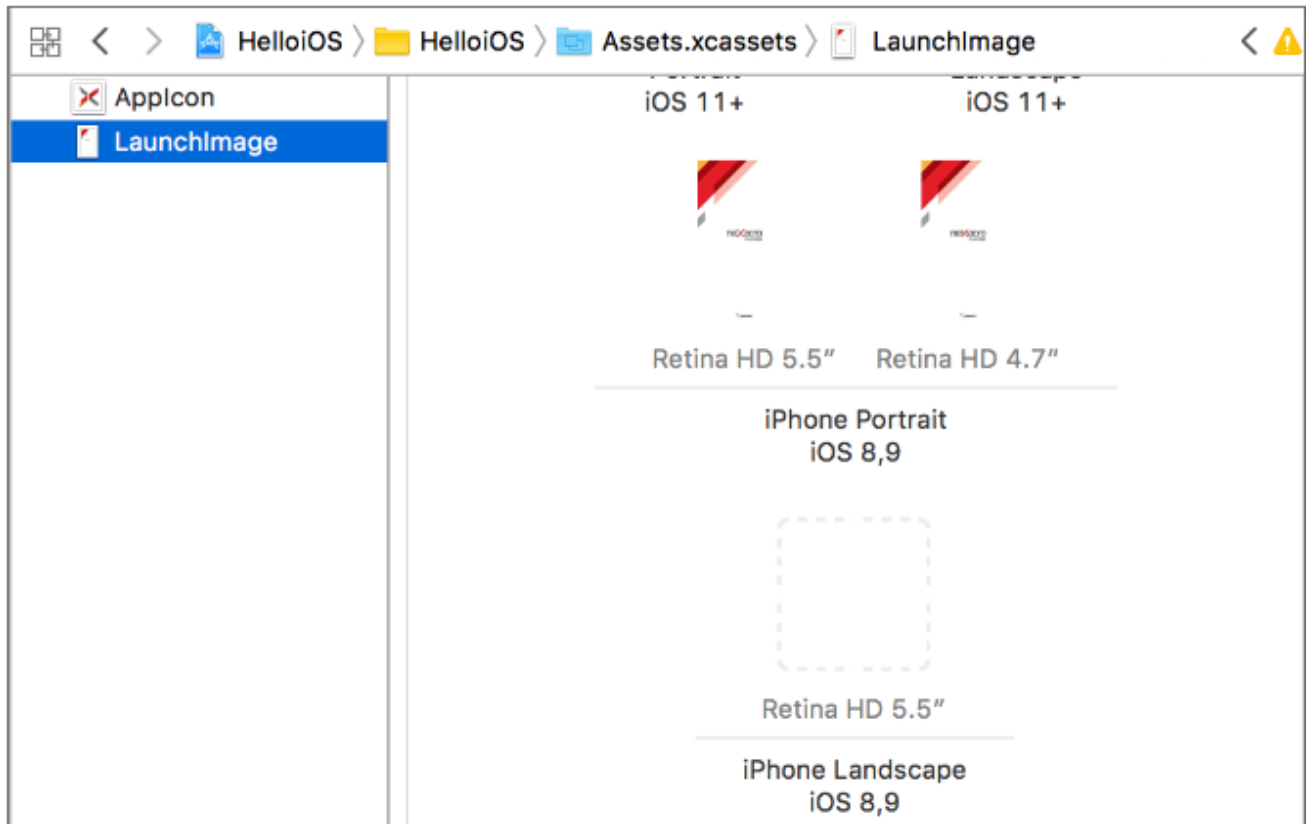


- 런치 이미지 설정

'Launch Images Source' 항목 옆의 'Use Asset Catalog...'를 클릭하면 다음과 같이 팝업이 표시됩니다. 템플릿으로 프로젝트를 생성했다면 기본 애셋 카탈로그가 이미 생성되어 있으므로 Assets를 선택한 후 Migrate 버튼을 클릭합니다.



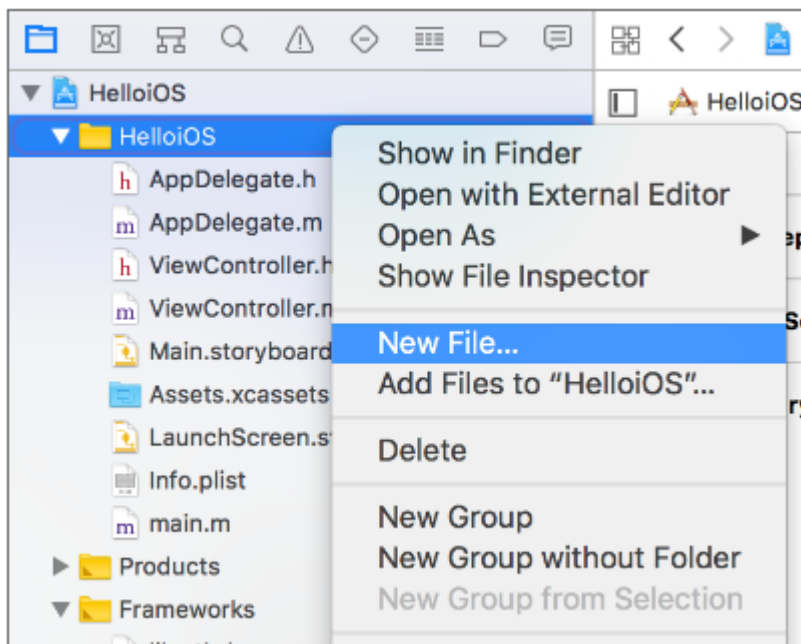
Migration이 수행되면 애셋 카탈로그에 새로 생성된 LaunchImage 항목이 보이고 그 옆으로 단말기와 용도에 맞게 설정할 수 있는 화면을 확인할 수 있습니다. 미리 제작한 런치 이미지를 알맞은 위치에 드래그 앤 드롭합니다.



## 메시지 설정

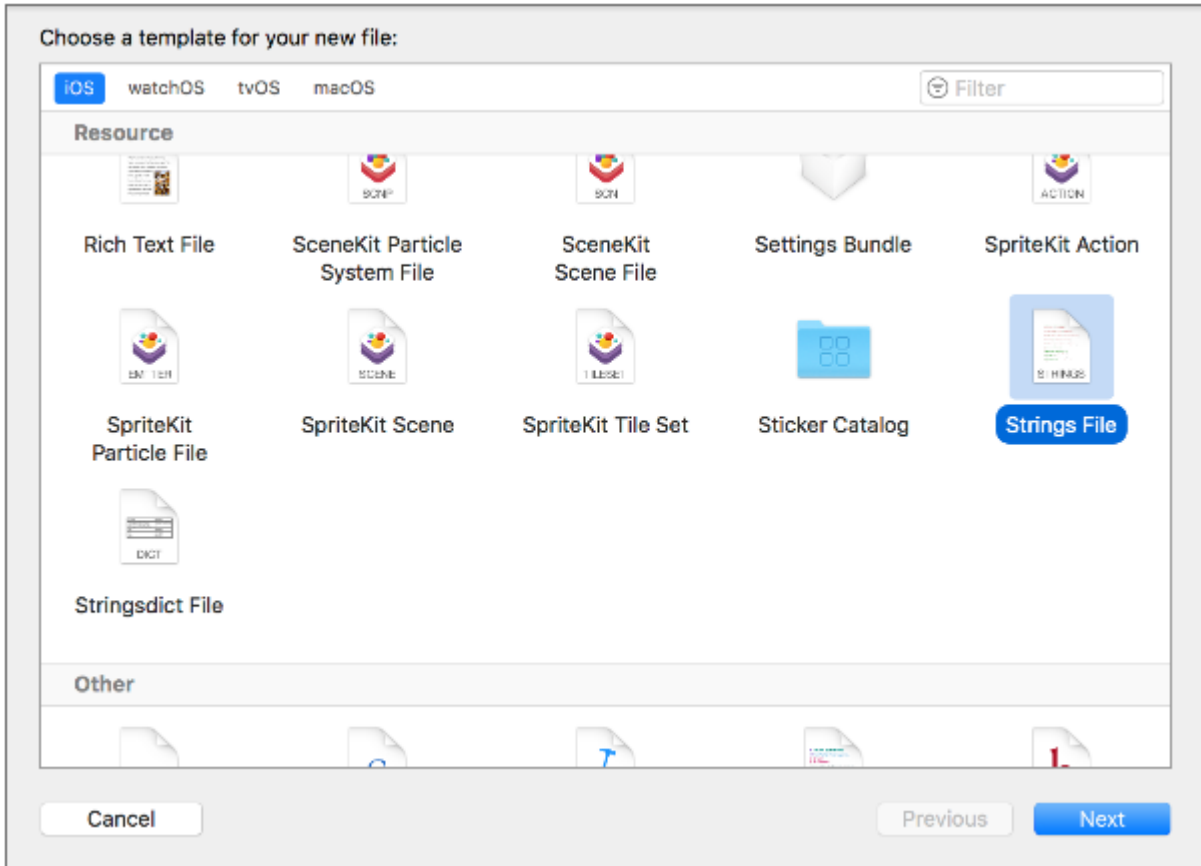
앱에서 사용할 메시지를 별도의 파일로 저장했다가 앱 실행 시 이용할 수 있습니다.

새로운 파일을 생성하려면 프로젝트 폴더를 선택 후 컨텍스트 메뉴에서 'New File'을 선택합니다.

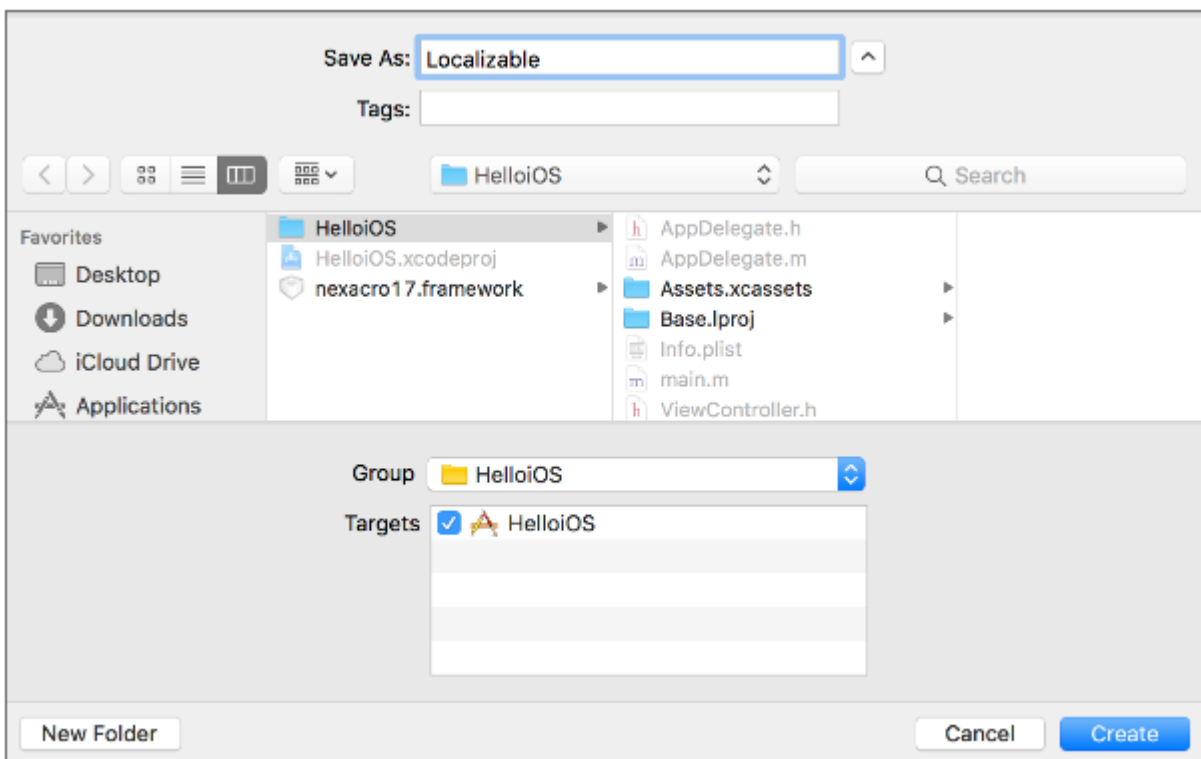


화면에 보이는 템플릿 중에서 'Strings File'을 선택합니다.





"Save As"에 파일명을 'Localizable'로 입력합니다.



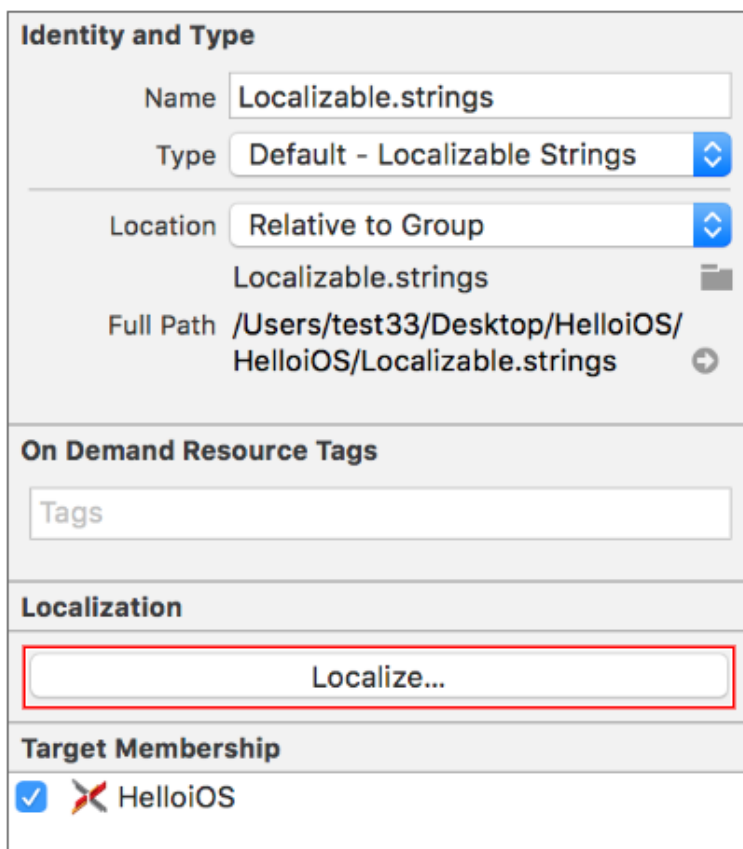
생성된 Localizable.strings 파일은 아래와 같이 수정합니다.

```

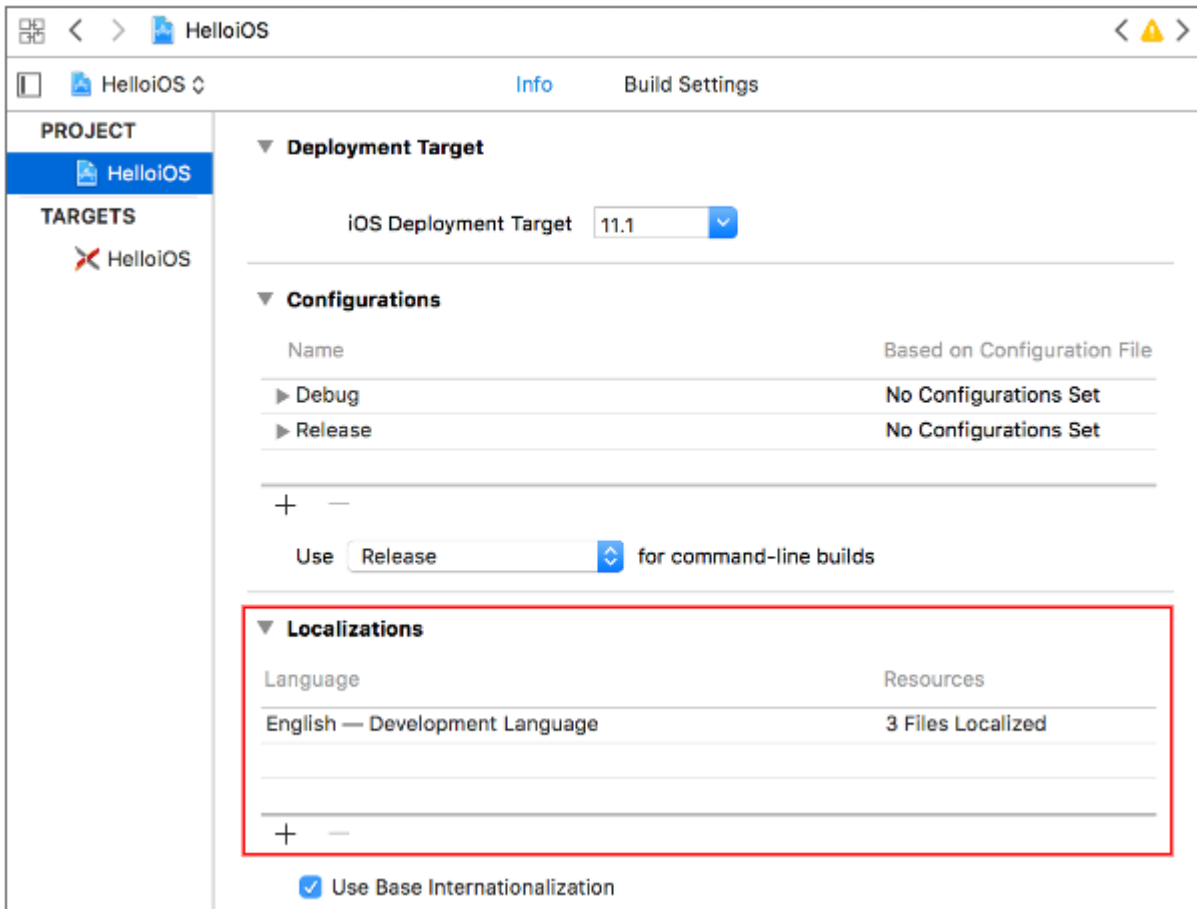
/*
    Localizable.strings
    HelloiOS
*/
"needupdate" = "It is need to update.";
"loadingFail" = "First loading is fail. \r\nPlease restart.";
"updateFail" = "Update is fail. \r\nPlease restart.";
"notexist" = "Start file is NOT exist. \r\nPlease restart.";
"BeingUpdated" = "Being updated.";
"wantreplace" = "There is a file of the save name. Do you want to replace? ";
"ok" = "OK";
"cancel" = "Cancel";
"move" = "Move";
"upper" = "Upper";
"filter" = "Filter";
"home" = "Home";
"nofilename" = "No File Name.";
"checkforupdates" = "Check for updates.";

```

Xcode 화면 오른쪽에 있는 Utilities 창에서 'Localize...' 버튼을 클릭하고 언어를 선택합니다. 사전에 프로젝트 설정에서 언어를 추가하지 않은 경우에는 기본으로 영어가 선택됩니다.



영어 이외의 언어를 지원하려면 [프로젝트 > Info > Localizations]에서 + 버튼을 눌러 다른 언어를 추가할 수 있습니다.



Localization에 관련된 자세한 내용은 애플 개발자 사이트를 참고하세요.

<https://developer.apple.com/library/content/documentation/MacOSX/Conceptual/BPInternational/Introduction/Introduction.html>

## 7.2.5 Config 설정

넥사크로플랫폼에서 제공하는 기능을 앱에서 사용할 수 있도록 nexacro\_config.xml 파일을 설정합니다. 앱 업데이트, Notification, 에러 정보 처리 등의 기능을 활성화 할 수 있습니다. nexacro\_config.xml 파일은 사용자가 직접 생성해야 하며 [{Asset} > data] 폴더 아래에 배치합니다. 이 파일은 Xcode에서 앱 빌드시 프로젝트에 포함되어 있어야 합니다.



nexacro\_config.xml 파일이 없으면 기본 설정이 적용되므로 앱 실행에는 아무런 지장이 없습니다.

nexacro\_config.xml 파일은 XML 형식이며 다음 예와 같이 작성합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<nexacro-config>
  <application dialog-position="top" use-wkwebview="true" file-logging="true" />
  <notification enable="true" handler="DefaultNotificationManagerHandler"/>
  <updater force="false" cancelable="false" quiet="true" errmsg="true"/>
  <xpush-server request-missing-message="true" />
</nexacro-config>
```

nexacro\_config.xml에 설정할 수 있는 기능은 다음과 같습니다.

기능	속성	설정 값	설명
application	dialog-position	"top"   "center"   "bottom"	앱 업데이트 진행 정보를 표시할 팝업 위치를 설정합니다.
	file-logging	"true"   "false"	로딩 에러 정보를 파일로 저장할지 설정합니다. iTunes를 통해서 로그 파일을 확인할 수 있습니다.
updater	force	"true"   "false"	start_ios.json에 업데이트 파일 정보 존재시 팝업으로 표시할지 설정합니다. "true"로 설정시 팝업 알림없이 강제로 업데이트를 진행합니다.
	cancelable	"true"   "false"	업데이트 파일 존재시 팝업에 업데이트 취소 버튼을 표시할지 설정합니다. 취소 버튼을 클릭하면 업데이트를 진행하지 않고 앱을 종료합니다.
	errmsg	"true"   "false"	앱 로딩에 실패했을 때 에러 정보를 팝업으로 표시할지 설정합니다.
	quiet	"true"   "false"	업데이트 팝업을 표시할지 설정합니다. "true"로 설정해도 업데이트 파일 존재시 진행 단계는 표시됩니다.
xpush-server	request-missing-message	"true"   "false"	xpush 서버 사용시 수신하지 못한 신뢰성 메시지를 자동으로 요청할지 설

기능	속성	설정 값	설명
			정합니다.
notification	enable	"true"   "false"	알림 기능을 사용할지 설정합니다.
	handler	"[클래스 이름]"	알림 수신시 처리할 핸들러를 구현한 클래스를 설정합니다. 기본 값은 "DefaultNotificationManagerHandler"입니다.

## 7.2.6 빌드 환경 설정

iOS 프로젝트를 생성하면서 기본으로 만들어진 AppDelegate.h, AppDelegate.m 파일과 main.m 파일을 넥사크로플랫폼 환경에 맞게 수정합니다.

### AppDelegate.h

생성된 애플리케이션 델리게이트가 넥사크로플랫폼 AppDelegate를 상속받도록 수정합니다. 파일을 다음과 같이 수정하여 nexacro17.framework가 동작할 수 있도록 합니다.

```
//
// AppDelegate.h
// HelloiOS
//

#import <UIKit/UIKit.h>
#import <nexacro17/NexacroAppDelegate.h>
#import <nexacro17/NexacroMainViewController.h>

@interface AppViewController : NexacroMainViewController
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation;
@end

@interface AppDelegate : NexacroAppDelegate <UIApplicationDelegate>
-(NexacroMainViewController*)initializeMainViewController;
@end
```

## AppDelegate.m

AppDelegate.m 파일에서는 넥사크로 스튜디오에서 만든 start\_ios.json 파일이 배치된 서버 URL 주소를 지정합니다. 아래 코드에서 굵게 표시된 부분을 사용자의 환경에 맞게 수정합니다.

```
//
// AppDelegate.m
// HelloiOS
//

#import "AppDelegate.h"

@implementation AppViewController
-(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    return [super shouldAutorotateToInterfaceOrientation:interfaceOrientation];
}
@end

@implementation AppDelegate
- (NexacroMainViewController*)initializeMainViewController
{
    NSString *bootstrapUrl = @"http://[URL]/start_ios.json";

    [[NexacroResourceManager sharedResourceManager] setBootstrapURL:bootstrapUrl isDirect:NO];
    AppViewController* controller = [[AppViewController alloc] initWithFullScreen:NO];

    return controller;
}
@end
```

## main.m

UIApplicationMain 함수는 UIKit 프레임워크에서 제공하는 기본 함수로 앱의 시작점입니다. 앱의 구동에 필요한 사전 준비를 하고 메인 루프를 시작해 사용자의 입력을 받을 준비를 합니다.

main.m 파일을 아래와 같은지 확인하고 다르다면 수정합니다.

```
//
// main.m
// HelloiOS
```

```
//

#import <UIKit/UIKit.h>
#import "AppDelegate.h"

int main(int argc, char * argv[]) {
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

## 기타 설정

'Build Settings' 항목에서 일부 설정을 수정해주어야 합니다.

- 'Architectures' 항목에서 'Build Active Architecture Only' 항목을 아래와 같이 수정합니다.

Build Settings > Architectures > Build Active Architecture Only

Debug: No

Release: No

- 'Apple LLVM 9.0 - Language - Object C' 항목에서 'Objective-C Automatic Reference Counting' 항목을 아래와 같이 수정합니다.

Build Settings > Apple LLVM 9.0 - Language - Object C > Objective-C Automatic Reference Counting

Yes

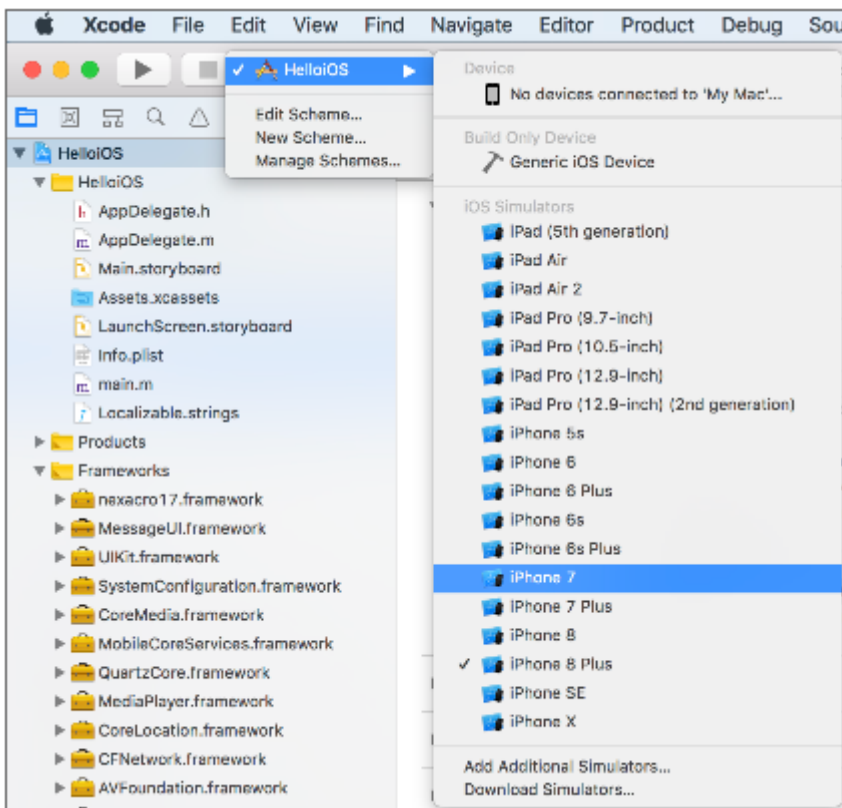
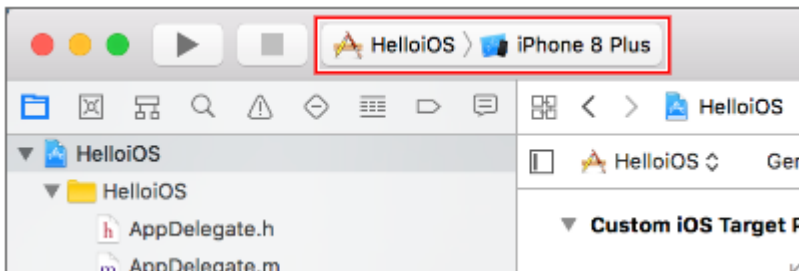
- Info.plist 파일을 선택하고 속성 목록에서 'View controller-based status bar appearance' 항목을 추가하고 값을 'NO'로 설정합니다.

Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(3 items)
View controller-based status...	Boolean	NO
		YES
		NO

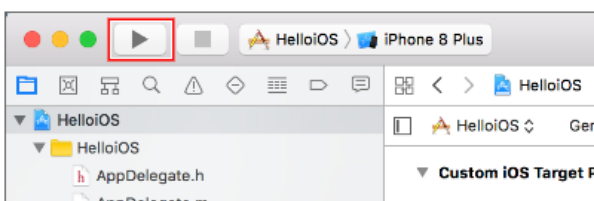
## 7.3 앱 테스트

iOS 단말기를 PC에 USB로 연결해 바로 테스트하거나 시뮬레이터에 연결해 테스트할 수 있습니다. 단말기를 직접 연결하는 경우에는 단말기에 적절한 프로파일이 설치되어 있어야 합니다.

1. Xcode 상단의 Target Name을 클릭하여 다음과 같이 앱을 구동시킬 iOS 단말기 혹은 시뮬레이터를 선택합니다.



2. Run 기능을 수행합니다. 화면 상단의 아이콘을 직접 클릭하거나 Xcode 메뉴에서 실행할 수 있습니다.





Product > Run



## 8.

# 앱 개발 및 실행 (안드로이드)

## 8.1 앱 개발 환경 설정

넥사크로플랫폼 안드로이드 앱은 전통적인 안드로이드 앱 개발 툴인 안드로이드 스튜디오를 이용하여 개발하거나 넥사크로플랫폼 제품군의 하나인 앱 빌더(App Builder)를 이용하여 개발할 수 있습니다. 이번 장에서는 안드로이드 스튜디오를 이용하여 넥사크로플랫폼 애플리케이션을 안드로이드 앱으로 개발하는 방법에 대해 설명합니다. 기본적인 개발 절차는 일반적인 안드로이드 앱을 만드는 방법과 동일합니다.

안드로이드 기본 개발 환경 구성은 다음과 같은 절차로 진행합니다.

1. JDK(Java SE Development Kit) 설치
2. 안드로이드 스튜디오 설치



이미 앱 개발 환경이 설정되어 있다면 앱 개발 환경 설정은 건너뛰어도 됩니다.

### 8.1.1 JDK(Java SE Development Kit) 설치

안드로이드 스튜디오는 자바 기반으로 만들어졌기 때문에 실행하기 위해서는 JDK(Java SE Development Kit) 환경이 설정되어 있어야 합니다. JDK는 무료로 제공되고 있으며 간단하게 설치할 수 있습니다.

## 내려받기

아래 사이트에서 시스템 환경에 맞는 JDK를 내려받을 수 있습니다.



<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



이번 장에서는 jdk-8u152-windows-i586.exe (version 1.8.0\_152-b16) 파일을 내려받아 사용했습니다. 사이트 업데이트에 따라 일부 내용이 변경될 수 있습니다.

## 설치 확인

내려받은 설치파일을 실행하면 자동으로 필요한 환경설정과 설치과정을 진행합니다. 설치가 완료되면 커멘드창에서 정상적으로 설치되었는지 확인합니다.

```
java -version
```

```
C:\>java -version
java version "1.8.0_152"
Java(TM) SE Runtime Environment (build 1.8.0_152-b16)
Java HotSpot(TM) Client VM (build 25.152-b16, mixed mode)
```



JDK 1.5 이상 버전을 설치하는 경우에는 실행에 필요한 환경 변수(JAVA\_HOME)를 따로 설정해주지 않아도 시스템 경로에 자동으로 실행 파일을 복사해 접근할 수 있습니다.

## 8.1.2 개발 도구 설치

안드로이드 스튜디오 설치파일은 기본적으로 SDK 도구를 포함하고 있습니다. 아래 사이트에서 환경에 맞는 설치파일을 내려받을 수 있습니다.

<https://developer.android.com/studio/index.html>



안드로이드 스튜디오를 기본 옵션으로 설치한 경우에는 바로 개발 단계로 넘어갑니다.



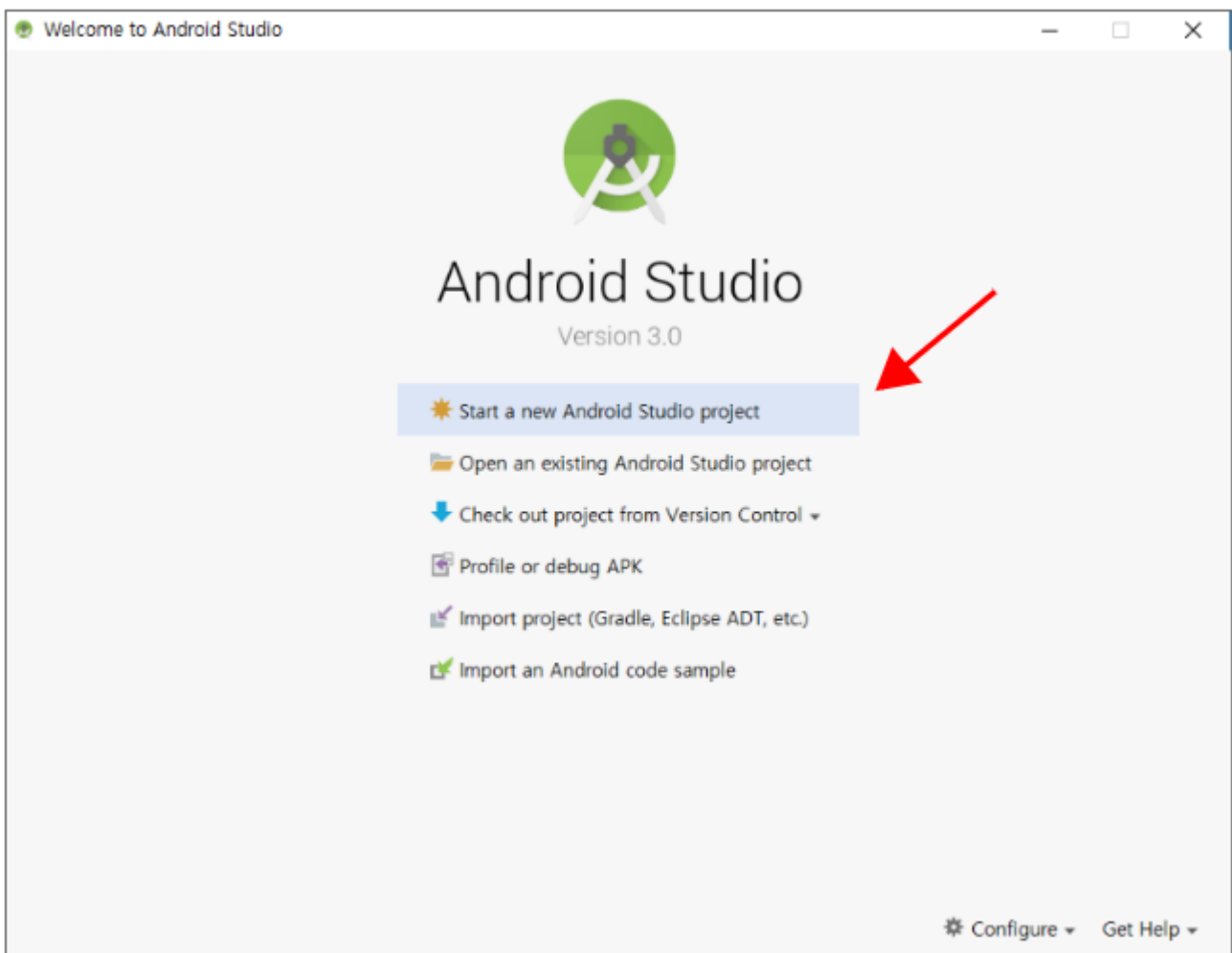
이번 장에서는 안드로이드 스튜디오 3.0 버전 설치를 기준으로 작성했습니다.

## 8.2 앱 프로젝트 개발

앱 개발 환경이 설정되었다면 안드로이드 스튜디오를 실행해서 안드로이드 운영체제에서 동작하는 앱을 만들 수 있습니다. 앱 프로젝트를 진행하기 전에 넥사크로 스튜디오에서 개발된 애플리케이션에서 만들어진 아카이브 파일은 지정된 경로에 위치해야 합니다.

### 8.2.1 프로젝트 생성

1. 기존에 열어놓은 프로젝트가 없다면 빠른 시작(Quick Start) 창에서 새로운 프로젝트를 생성합니다. 아래 그림에서 [Start a new Android Studio project]를 선택합니다.



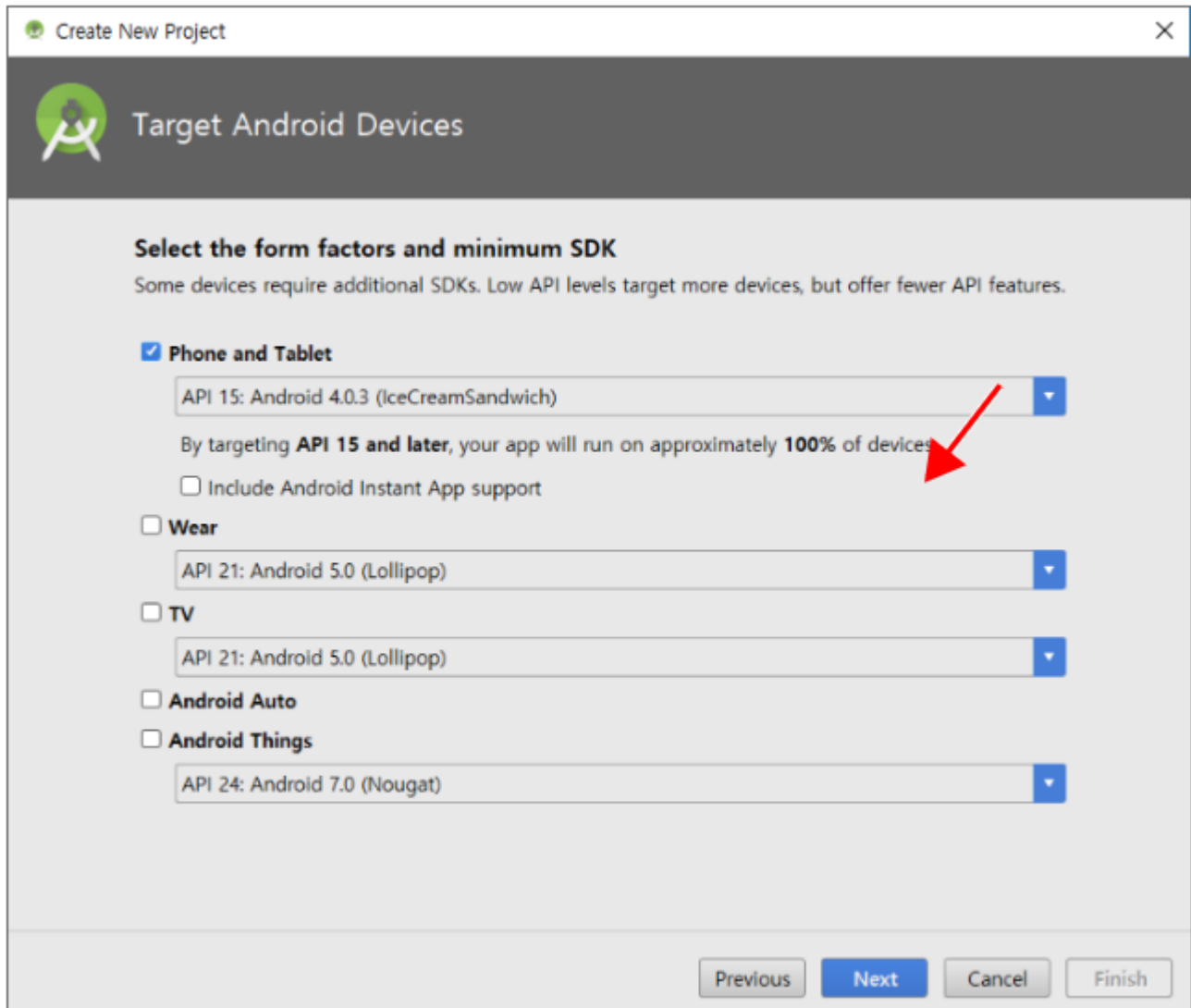
다른 프로젝트 작업 중에 새로운 프로젝트를 생성하고자 한다면 아래 메뉴에서 생성할 수 있습니다.

File > New > New Project

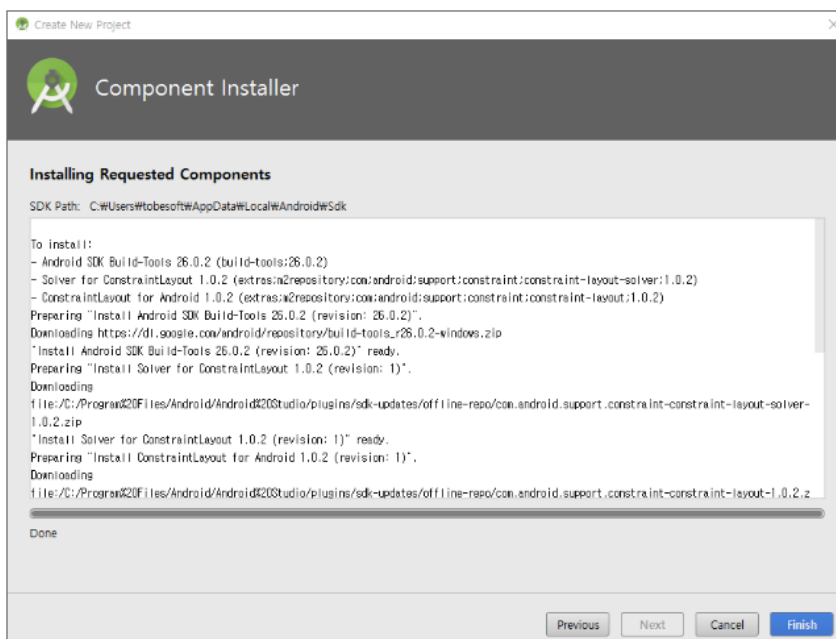
2. 앱의 이름, 도메인 그리고 프로젝트의 경로를 설정합니다.

	항목	설명
①	Application name	생성할 앱 이름을 입력합니다.
②	Company domain	회사 도메인명을 입력하면 패키지 이름을 자동으로 생성합니다.
③	Project location	프로젝트가 저장될 위치를 지정합니다.
④	Package Name	입력한 회사 도메인명을 기준으로 패키지 이름을 생성합니다.

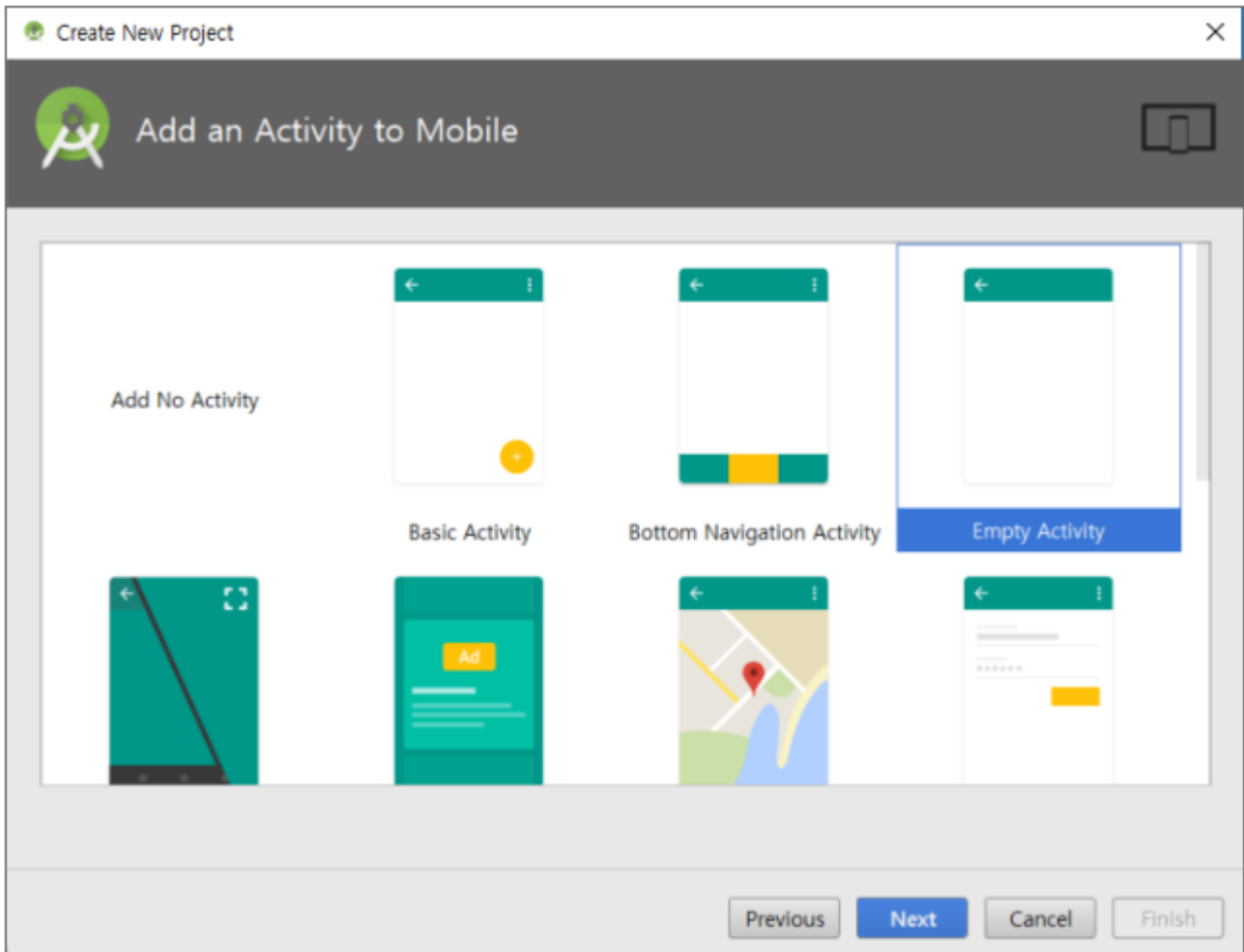
3. 안드로이드 디바이스 종류 및 최소 사양 SDK를 설정합니다. SDK를 변경할 때마다 항목 하단에 얼마나 많은 디바이스와의 호환성을 제공하는지에 대한 정보를 제공합니다.



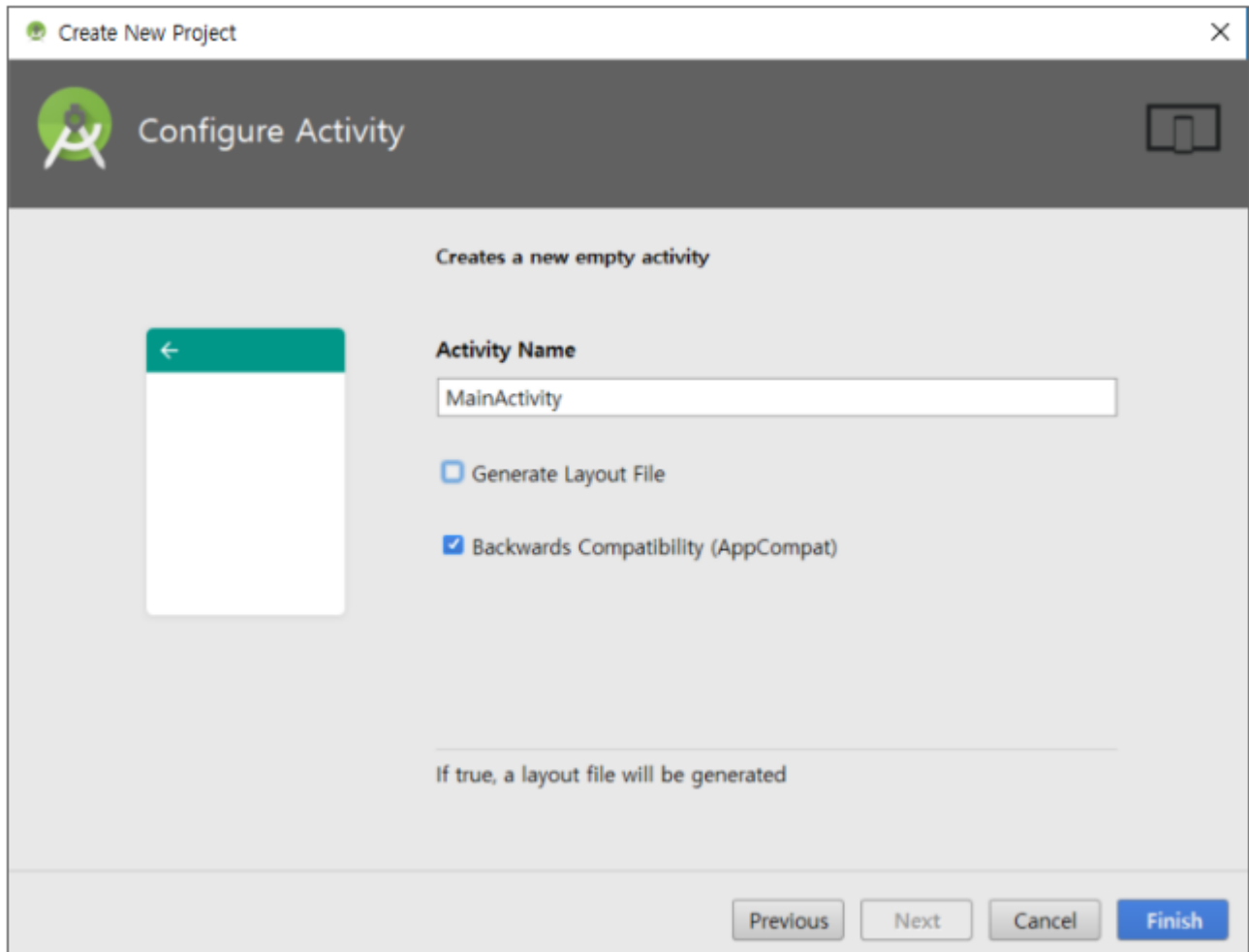
단계 진행중에 필요한 버전의 SDK가 설치되어 있지 않을 경우에는 자동으로 SDK를 내려받아 설치합니다.



4. 안드로이드에서 제공하는 프로젝트 템플릿을 선택합니다. 선택하는 템플릿에 따라 구조, 화면 레이아웃, 파일 등의 구성이 달라집니다. 여기서는 넥사크로 애플리케이션을 담기 위한 기본적인 앱 형태면 충분하므로 Empty Activity를 선택한 다음 [Next] 버튼을 클릭합니다.



5. 액티비티 설정 화면에서 액티비티 이름을 입력하고, Generate Layout File 체크를 해제한 후 [Finish] 버튼을 클릭하면 새로운 프로젝트가 생성됩니다.

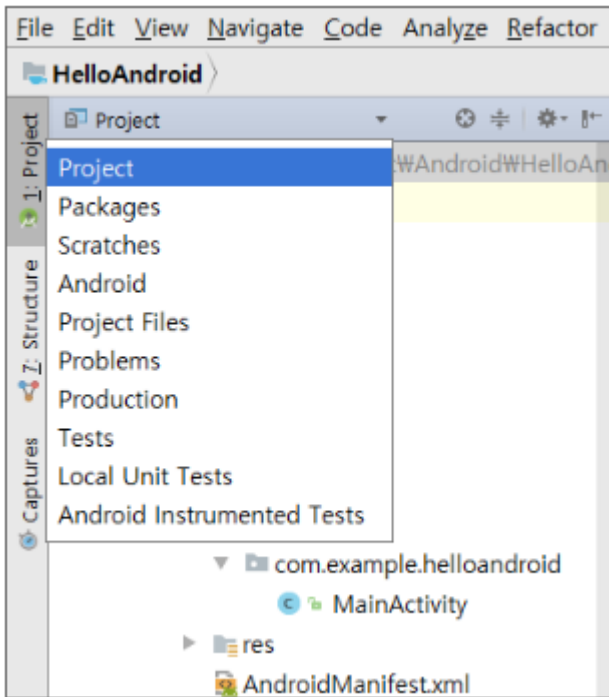


## 8.2.2 넥사크로플랫폼 라이브러리 설정

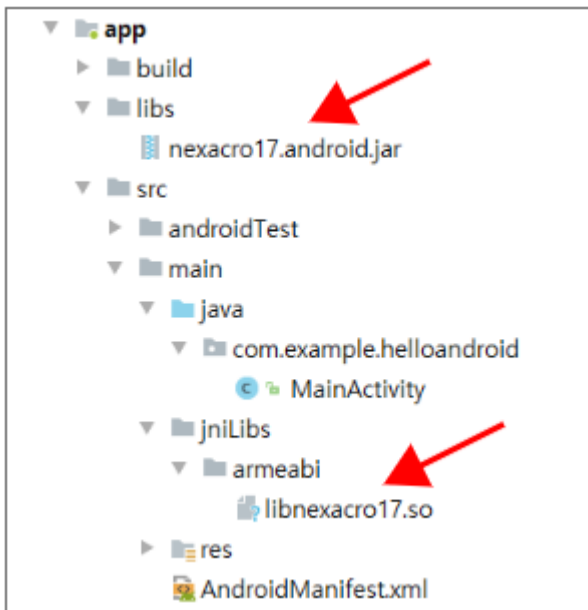
안드로이드 프로젝트에서 넥사크로플랫폼 애플리케이션을 앱으로 만들기 위해 추가로 제공되는 넥사크로플랫폼 라이브러리 파일(nexacro17.android.jar, libnexacro17.so)을 애플리케이션에서 사용할 수 있도록 설정해야 합니다.

라이브러리 파일을 실제 프로젝트 디렉토리의 정해진 위치에 복사하려면 프로젝트 윈도우 상단에서 뷰 옵션을 [Project]로 변경합니다. 안드로이드 스튜디오를 설치하면 기본적으로 뷰 모드가 [Android]로 되어 있는데 이를 [Project]로 설정해야 모든 디렉토리 구조를 확인할 수 있습니다.

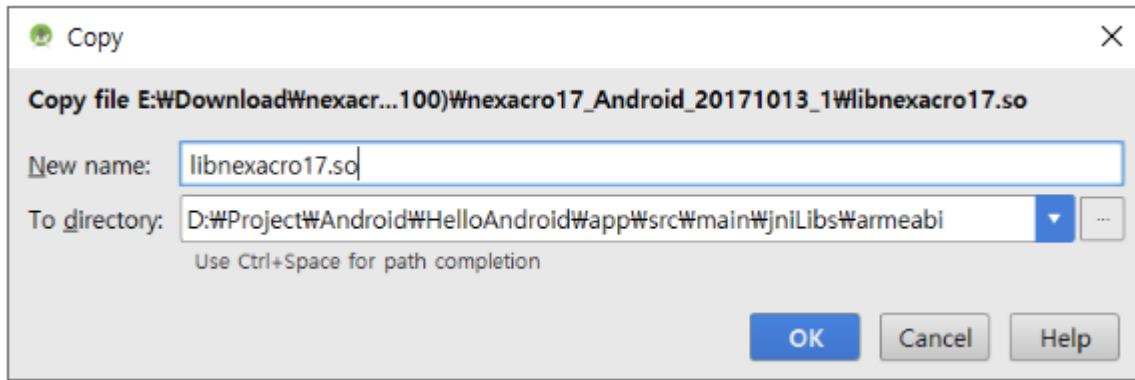




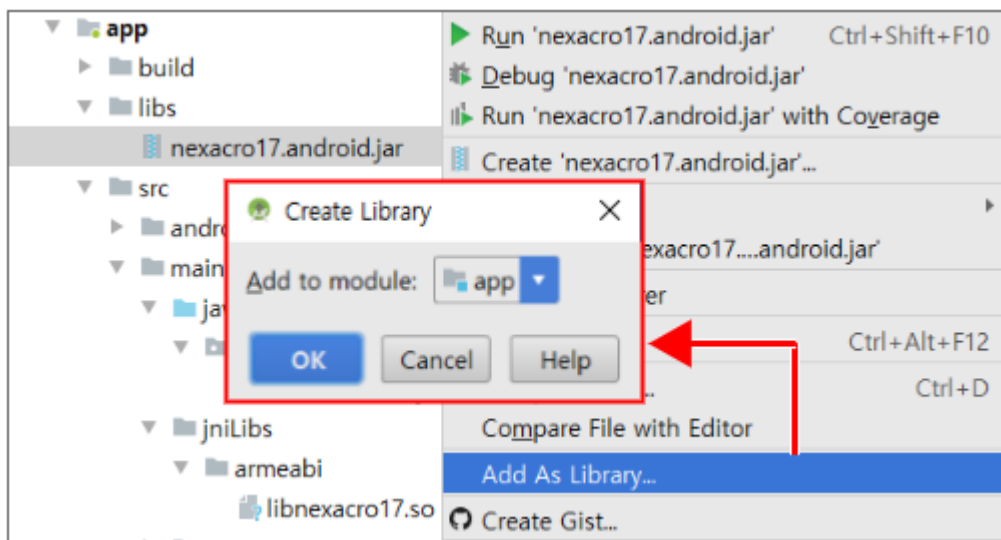
nexacro17.android.jar 파일은 [프로젝트 > app > libs] 폴더로 복사하고 libnexacro17.so 파일은 [프로젝트 > app > src > main] 폴더 아래 [jniLibs > armeabi] 폴더를 새로 만들어 복사합니다. 생성된 폴더 구조는 아래 그림과 같습니다.



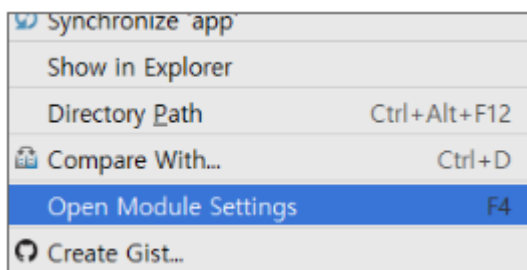
파일을 추가할 때 아래 그림처럼 파일명과 경로를 변경할 수 있는 창이 나타납니다. 기본 설정된 값으로 [OK] 버튼을 클릭합니다.



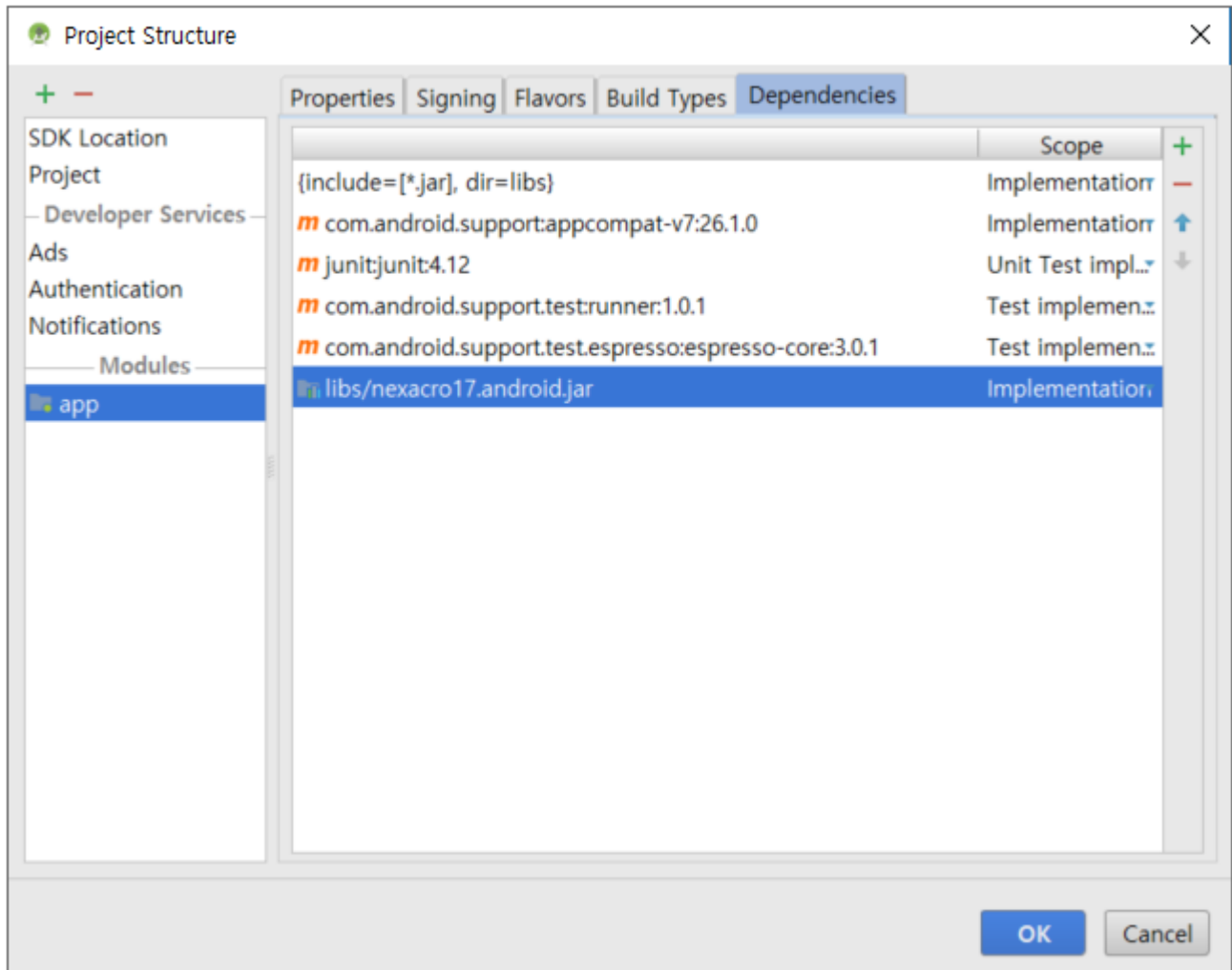
추가된 넥사크로플랫폼 라이브러리를 프로젝트에서 참조하도록 라이브러리에 추가합니다. nexacro17.android.jar 파일을 선택하고 컨텍스트 메뉴에서 [Add As Library] 항목을 선택합니다.



라이브러리가 정상적으로 추가되었는지 확인하려면 [프로젝트 > app] 항목을 선택하고 컨텍스트 메뉴에서 [Open Module Settings] 항목을 선택합니다.



[Modules > app] 항목에서 상단 탭을 [Dependencies]로 변경하면 라이브러리가 추가된 것을 확인할 수 있습니다.

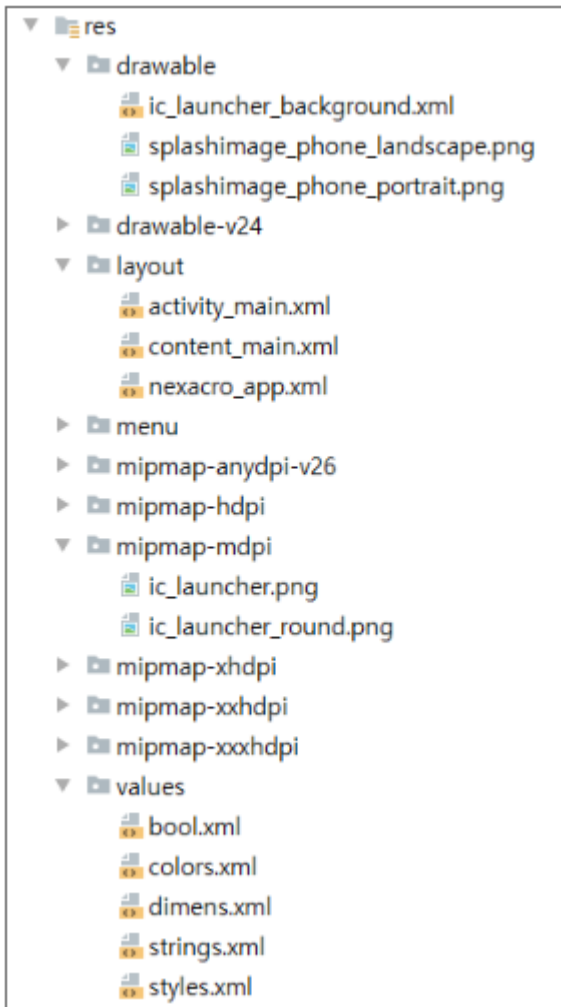


### 8.2.3 리소스 설정

앱에서 사용할 로딩 이미지, 아이콘, 메시지, 레이아웃 등을 설정하는 단계입니다. 리소스 파일을 단말기의 화면 크기, 밀도에 따라 만든 후 적절한 폴더에 배치하면 안드로이드 시스템은 사용자에게 최적화된 뷰를 제공합니다.

안드로이드 시스템은 단말기의 화면이나 밀도에 따라 적절한 리소스를 판단해 사용합니다. 그러나 단말기 화면에 맞는 리소스가 없을 경우에는 기본 리소스를 화면에 맞게 확대/축소하여 사용합니다.

리소스 폴더는 프로젝트 생성시 자동으로 생성되며 일반적으로 아래와 같은 구조를 가집니다. 개발자는 리소스의 종류에 따라 적절한 폴더에 이미지 및 XML 파일을 배치합니다.



리소스 폴더는 <리소스 명>-<구성 한정자> 형태로 구성됩니다. 구성 한정자(Configuration Qualifier)는 종류에 따라 크기(small, normal, large..), 밀도(ldpi, mdpi, hdpi, xhdpi..), 방향(land, port), 화면비(long, notlong) 등이 될 수 있습니다. 예를 들어, drawable-xxhdpi는 초고밀도(Extra High Density) 이미지와 XML 리소스 파일을 배치하는 폴더입니다. drawable과 같이 구성 한정자가 붙어 있지 않은 폴더는 시스템에서 화면을 그릴 때 기준이 되는 기본 리소스를 배치하는 폴더입니다.

리소스에 관한 더 상세한 내용은 안드로이드 매뉴얼을 참조하시기 바랍니다.



<https://developer.android.com/guide/topics/resources/overview.html>

## 이미지 설정

앱에서 사용하는 이미지는 res 폴더 아래에 밀도에 따라 알맞은 폴더에 지정합니다. 예를 들어, 로딩 이미지는 drawable 폴더로, 앱 아이콘은 mipmap 폴더로 지정합니다. 적용할 이미지 파일은 앱에서 지원하는 해상도에 따라 폴더를 달리 설정해야 합니다. 상세한 내용은 안드로이드 매뉴얼을 참고해주세요.



[http://developer.android.com/guide/practices/screens\\_support.html](http://developer.android.com/guide/practices/screens_support.html)

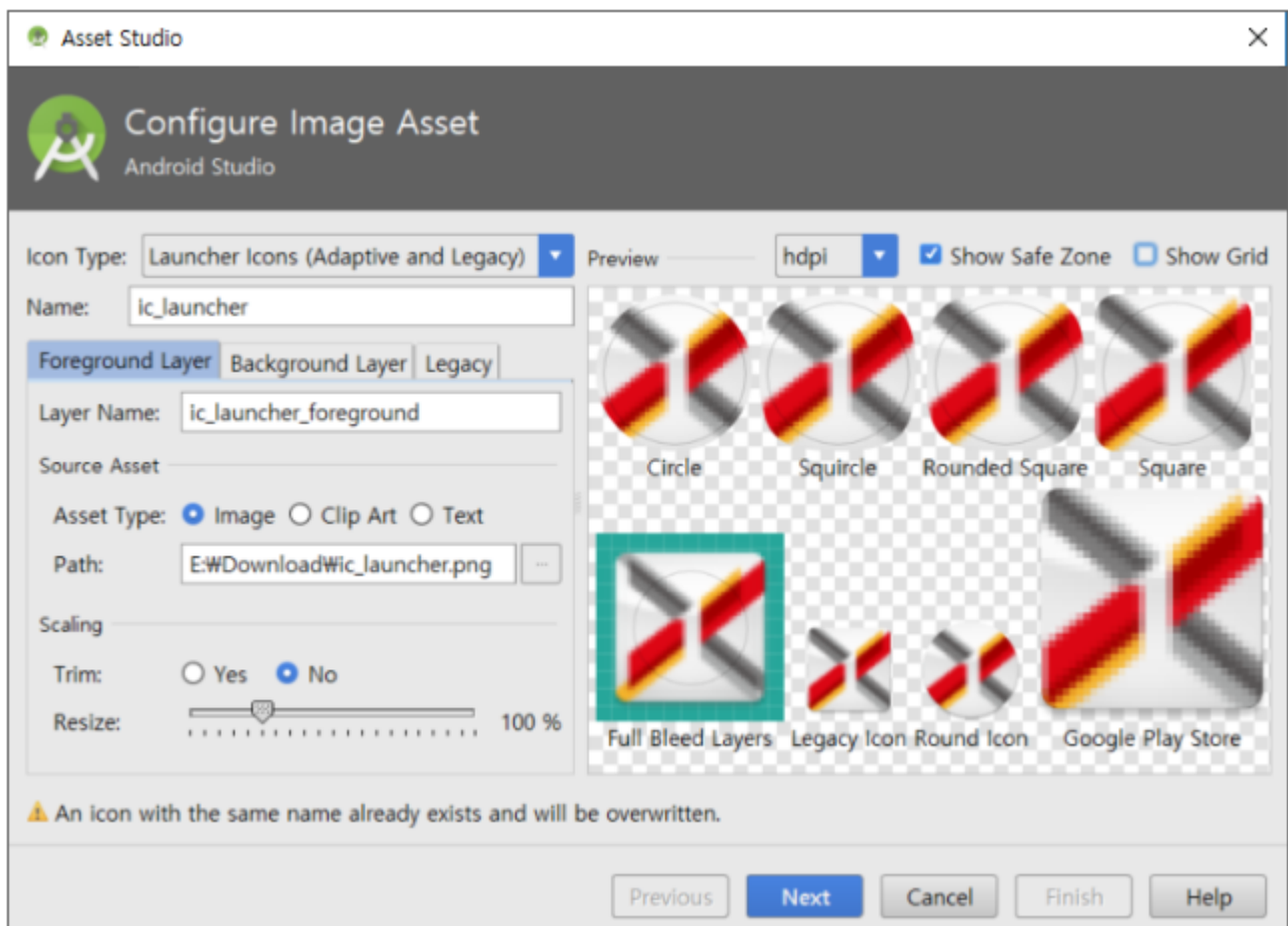
항목	설명
ic_launcher.png	바탕화면에 설치되는 애플리케이션 아이콘 이미지입니다. AndroidManifest.xml 파일에서 파일명을 변경할 수 있습니다.
splashimage_phone_landscape.png	안드로이드 폰 가로 방향 스플래시 이미지입니다(모두 소문자).
splashimage_phone_portrait.png	안드로이드 폰 세로 방향 스플래시 이미지입니다(모두 소문자).
splashimage_pad_landscape.png	안드로이드 태블릿 가로 방향 스플래시 이미지입니다(모두 소문자).
splashimage_pad_portrait.png	안드로이드 태블릿 세로 방향 스플래시 이미지입니다(모두 소문자).

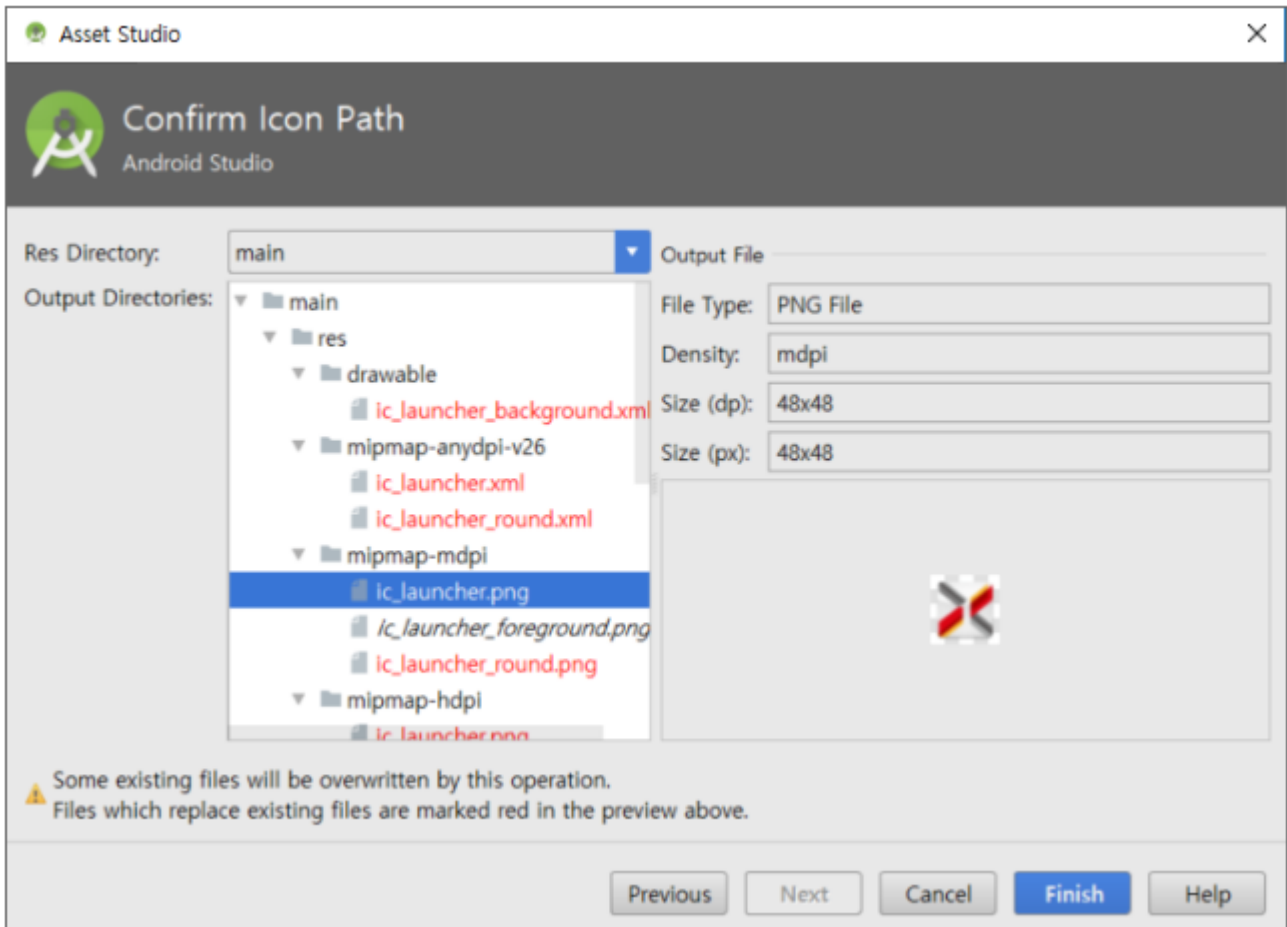


런처 아이콘, 로딩 이미지 등을 별도로 지정하지 않는 경우에는 기본 설정이 적용되므로 앱 동작에는 아무런 문제가 없습니다.

- 아이콘 이미지 설정

아이콘 파일은 프로젝트 생성 후 수정할 수 있습니다. [프로젝트 > app] 항목 선택 후 컨텍스트 메뉴에서 [New > Image Asset] 항목을 선택한 후 나타난 [Asset Studio]에서 아이콘 파일 관련 설정을 변경합니다.

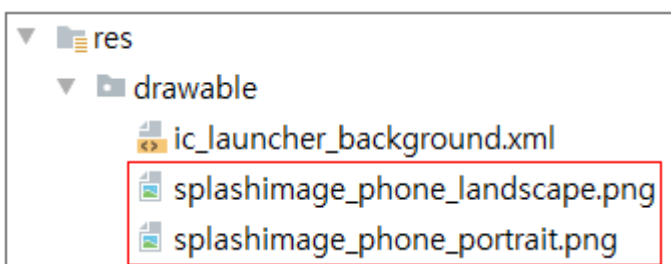




<http://developer.android.com/intl/ko/tools/help/image-asset-studio.html>

- 로딩 화면(스플래시) 설정

애플리케이션이 구동될 때 보여지는 로딩 화면을 간단히 설정할 수 있습니다. 위 표의 'splashimage\_'로 시작하는 파일이 넥사크로 애플리케이션에서 사용하는 로딩 이미지입니다. 사용자는 로딩 화면으로 사용할 이미지를 /res/drawable 폴더에 복사한 후 파일명 다음과 같이 변경해 줍니다.



## 문자열 설정

앱 실행 시 사용할 문자열을 설정합니다. 프로젝트 생성 시 만들어지는 res 폴더 아래 values 폴더에 있는 strings.xml 파일을 아래와 같이 수정합니다.



아래는 넥사크로 앱에서 사용하는 기본 문자열입니다. 문자열은 사용자의 상황에 맞게 수정할 수 있으나 문자열 name은 앱에서 참조하여 사용하므로 수정하지 않습니다. 넥사크로 앱의 기본 문자열을 설정하지 않으면 앱이 정상적으로 동작하지 않을 수 있습니다.

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- app_name은 실제 디바이스에 설치되는 앱의 이름을 정의합니다. 이를 수정하면 화면에
    표시되는 앱의 이름을 변경할 수 있습니다. -->
    <string name="app_name">HelloAndroid</string>
    <string name="action_settings">Settings</string>

    <!-- Clipboard -->
    <string name="copy">Copy</string>
    <string name="paste">Paste</string>
    <string name="cut">Cut</string>
    <string name="close">Close</string>
    <string name="selectall">Select All</string>
    <string name="selectword">Select Word</string>

    <string name="needupdate">It is need to update. \nAfter completing, Please restart.</string>
<
    <string name="loadingFail">First loading is fail. \nPlease restart.</string>
    <string name="updateFail">Update is fail. \nPlease restart.</string>
    <string name="notexist">Start file is NOT exist. \nPlease restart.</string>
    <string name="BeingUpdated">Being updated.</string>
    <string name="wantreplace">There is a file of the save name. Do you want to replace? </
string>
    <string name="ok">OK</string>
    <string name="cancel">Cancel</string>
    <string name="move">Move</string>
    <string name="upper">Upper</string>
    <string name="filter">Filter</string>
    <string name="home">Home</string>
    <string name="nofilename">No File Name.</string>
    <string name="checkforupdates">Check for updates.</string>
    <string name="installforupdates">Install for updates.</string>
    <string name="downloadingforupdates">Downloading for updates.</string>
    <string name="force_close">Update is Completed.</string>
    <string name="LoadingPleaseWait">Loading... \nPlease Wait...</string>
```

```

<!-- Accessibility widget role -->
<string name="alert">alert</string>
<string name="pushbutton">pushbutton</string>
<string name="combobox">combobox</string>
<string name="checkboxbutton">checkboxbutton</string>
<string name="radiobutton">radiobutton</string>
<string name="text">text</string>
<string name="list">list</string>
<string name="listitem">listitem</string>
<string name="statictext"></string>
<string name="graphic">image</string>
<string name="spacebar">space bar</string>
<string name="textdeleted">deleted</string>

<!-- Accessibility widget statue -->
<string name="checked">checked</string>
<string name="unchecked">unchecked</string>

<!-- Accessibility reaction -->
<string name="click">click</string>

<!-- Splash message -->
<string name="start_initialize">Start Initialize</string>
<string name="check_license">Check License</string>
<string name="load_frameworkmodule">Load Framework Module</string>
<string name="load_extendmodule">Load Extend Module</string>
<string name="execute_frameworkscript">Execute Framework Script</string>
<string name="load_application">Load Application</string>
<string name="load_failapplication">Load Fail Application Initialize. Click And Close</string>

</resources>

```

문자열은 사용자의 상황에 맞게 수정할 수 있습니다. 그러나 위에 정의된 문자열 name은 앱에서 고정적으로 사용하므로 수정하지 않습니다.

영문 문자열 외 한국어나 일본어 등의 문자열을 추가하려면 res 폴더 아래 values-ko, values-ja 와 같은 식으로 폴더를 추가하고 strings.xml 파일을 추가합니다. 더 상세한 내용은 안드로이드 매뉴얼을 참조하시기 바랍니다.





<https://developer.android.com/training/basics/supporting-devices/languages.html?hl=ko>

## 레이아웃 설정

단말기에서 보이는 앱의 레이아웃을 지정합니다. 안드로이드 앱에서 넥사크로플랫폼 애플리케이션을 감싸기 위한 용도이므로 기본적인 상태로 설정합니다. 레이아웃을 설정하지 않으면 앱 실행시 넥사크로플랫폼 애플리케이션이 정상적으로 화면에 출력되지 않습니다.

res 폴더 아래에 있는 layout 폴더에 nexacro\_app.xml 파일을 새로 만들어 아래의 같이 수정합니다. 레이아웃 파일을 새로 추가하려면 [File > New > XML > Layout XML File]을 선택합니다.

nexacro\_app.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- nexacro_app.xml -->
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nexacro_activity"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/transparent"
    tools:context="com.nexacro.deviceAPI.FileDialogActivity" >

    <com.nexacro.view.NexacroLayout
        android:id="@+id/nexacro_layout"
        android:background="@android:color/transparent"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </com.nexacro.view.NexacroLayout>

</FrameLayout>
```



레이아웃을 설정하지 않으면 화면이 정상적으로 표시되지 않거나 에러가 발생할 수 있습니다.

## 8.2.4 Config 설정

넥사크로플랫폼에서 제공하는 기능을 앱에서 사용할 수 있도록 nexacro\_config.xml 파일을 설정합니다. 앱 업데이트, Notification, 에러 정보 처리 등의 기능을 활성화 할 수 있습니다. nexacro\_config.xml 파일은 사용자가 직접 생

성해야 하며 [프로젝트 > app > src > main > res > xml] 폴더 아래에 배치합니다. 이 파일은 안드로이드 스튜디오에서 앱 빌드시 프로젝트에 포함되어 있어야 합니다.



nexacro\_config.xml 파일이 없으면 기본 설정이 적용되므로 앱 실행에는 아무런 지장이 없습니다.

nexacro\_config.xml 파일은 XML 형식이며 다음 예와 같이 작성합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<nexacro-config>
  <application dialog-position="center" file-logging="true" quiet="false"/>
  <updater cancelable="true" force="true" errormsg="true" quiet="false"/>
  <xpush-server requestMissingMessage="true"/>
  <notification enable="true" project-number="336606568255" ui-handler="com.nexacro.
notification.DefaultUIHandler" handler="com.nexacro.notification.DefaultHandler"/>
</nexacro-config>
```

nexacro\_config.xml에 설정할 수 있는 기능은 다음과 같습니다.

기능	속성	설정 값	설명
application	dialog-position	"top"   "center"   "bottom"	앱 업데이트 진행 정보를 표시할 팝업 위치를 설정합니다.
	file-logging	"true"   "false"	로딩 에러 정보를 파일로 저장할지 설정합니다. "true"로 설정시 먼저 외부 저장소(context.getExternalCacheDir())로 저장하고, 설정된 경로가 없으면 내부 저장소(context.getCachePath())에 저장합니다. 파일은 [저장소경로]/logs/yyyy_MM_dd.txt 형태로 저장됩니다.
	quiet	"true"   "false"	앱 실행 팝업을 표시할지 설정합니다. "true"로 설정시 "Loading Application" 실행 메시지가 출력되지 않습니다.
updater	force	"true"   "false"	start_android.json에 업데이트 파일 정보 존재시 팝업으로 표시할지 설정함

기능	속성	설정 값	설명
			니다. "true"로 설정시 팝업 알 림없이 강제로 업데이트를 진행합니다.
	cancelable	"true"   "false"	업데이트 파일 존재시 팝 업에 업데이트 취소 버튼 을 표시할지 설정합니다. 취소 버튼을 클릭하면 업 데이트를 진행하지 않고 앱을 실행합니다.
	errmsg	"true"   "false"	앱 로딩에 실패했을 때 에 러 정보를 팝업으로 표시 할지 설정합니다.
	quiet	"true"   "false"	업데이트 팝업을 표시할지 설정합니다. "true"로 설정해도 업데이 트 파일 존재시 진행 단계 는 표시됩니다.
xpush-server	requestMissingMessage	"true"   "false"	xpush 서버로 미수신 메 시지를 자동으로 요청할지 설정합니다.
notification	enable	"true"   "false"	알림기능을 사용할지 설정 합니다.
	project-number	구글 GCM 프로젝트 번호	알림 기능용 구글 GCM 프로젝트 번호를 설정합니 다.
	ui-handler	"[함수 이름]"	알림 수신시 UI 관련 처리 를 위한 함수를 설정합니 다. 기본 값은 "com.nexacro. notification.DefaultUIH andle" 입니다.
	handler	"[함수 이름]"	알림 수신시 메시지 혹은 데이터 처리를 위한 함수 를 설정합니다. 기본 값은 "com.nexacro. notification.DefaultHan dler" 입니다.

## 8.2.5 빌드 환경 설정

안드로이드 프로젝트를 생성하면서 기본으로 만들어진 MainActivity.java 파일과 AndroidManifest.xml 파일을 넥사크로플랫폼 환경에 맞게 수정합니다.

### MainActivity.java

프로젝트 생성 시 액티비티 이름을 지정해주는데 별도로 수정하지 않는다면 MainActivity.java 라는 이름으로 생성됩니다. [프로젝트 > app > src > main > java] 폴더에서 지정된 패키지 이름 아래에 MainActivity.java 파일을 선택하고 아래와 같이 수정해줍니다.

MainActivity.java

```
package com.example.helloandroid;

import com.nexacro.NexacroResourceManager;
import com.nexacro.NexacroUpdaterActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends NexacroUpdaterActivity {

    public MainActivity() {
        super();

        setBootstrapURL("http://[URL]/start_android.json");
        setProjectURL("http://[URL]/");
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        NexacroResourceManager.createInstance(this);
        NexacroResourceManager.getInstance().setDirect(false);

        Intent intent = getIntent();
        if(intent != null) {
            String bootstrapURL = intent.getStringExtra("bootstrapURL");
            String projectUrl = intent.getStringExtra("projectUrl");
```

```

        if(bootstrapURL != null) {
            setBootstrapURL(bootstrapURL);
            setProjectURL(projectUrl);
        }
    }

    super.onCreate(savedInstanceState);
}

@Override
public void setContentView(View view) {
    super.setContentView(view);
}
}

```

위의 코드에서 굵은 글씨로 표시된 부분은 사용자가 환경에 맞게 수정해야 합니다.

15~16번 줄은 부트스트랩 주소와 프로젝트 소스의 주소를 설정하는 부분입니다. setBootstrapURL 메소드의 인수는 넥사크로 스튜디오에서 만든 start\_android.json 파일이 배치된 서버 URL을 지정하고 setProjectURL 메소드의 인수는 해당 프로젝트 소스가 제너레이트된 경로를 지정합니다. 제너레이트 소스의 경로 끝에는 '/'를 붙여줍니다.

```

15 setBootstrapURL("http://192.168.0.1:8080/nexacro/_android/start_android.json");
16 setProjectURL("http://192.168.0.1:8080/nexacro/_android/");

```

22~23번 줄은 애플리케이션의 Update Type에 따라 수정합니다. Update Type은 넥사크로 스튜디오에서 Packing (Archive&Update)시 설정하는데 Update Type이 'Server'인 경우에는 23번 줄의 setDirect 메소드의 인수를 아래와 같이 'true'로 설정합니다. Update Type이 'Update(Local+Server)' 혹은 'Local' 타입인 경우에는 'false'로 설정합니다.

```

22 NexacroResourceManager.createInstance(this);
23 NexacroResourceManager.getInstance().setDirect(true);

```



부트스트랩 URL, 프로젝트 URL 그리고 리소스 매니저를 잘못 설정하면 앱이 정상적으로 동작하지 않을 수 있으니 유의하십시오.

## AndroidManifest.xml

AndroidManifest.xml 파일은 [프로젝트 > app > src > main]에 위치하며 애플리케이션에 대한 기본적인 정보와 특정 기능 실행 시 필요한 정보를 담고 있습니다. 기본적으로 필요한 정보는 아래 코드를 참고해 작성합니다. 아래 내용 중 굵게 표시된 package, MainActivity 에 대한 정보는 프로젝트 설정에 따라 지정해야 합니다.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloandroid">

    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="19" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <!-- network access Permission -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />

    <!-- storage access permission -->
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_INTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name="com.example.helloandroid.MainActivity"
```

```

        android:label="@string/app_name"
        android:theme="@android:style/Theme.NoTitleBar">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <activity
        android:name="com.nexacro.NexacroActivity"
        android:alwaysRetainTaskState="true"
        android:configChanges="orientation|keyboardHidden|screenSize"
        android:launchMode="singleTop"
        android:theme="@android:style/Theme.NoTitleBar"
        android:windowSoftInputMode="adjustResize" >
    </activity>

    <!-- FileDialog -->
    <activity android:name="com.nexacro.deviceAPI.FileDialogActivity" android:
screenOrientation="sensor">
        <intent-filter>
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="google_play_services_version" />

</application>

</manifest>

```

배포할 앱의 기능에 따라 필요한 권한 및 기타 정보를 추가로 지정합니다. 예를 들어, 카메라 기능을 사용한다면 아래와 같은 항목이 추가되어야 합니다.

```

...
<uses-permission android:name="android.permission.CAMERA" />
<activity android:name="com.nexacro.deviceAPI.CameraListener"
    android:screenOrientation="landscape">
    <intent-filter>

```

```

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus"
    android:required="false"/>
...

```

다음은 많이 사용되는 기능의 권한 설정 코드입니다. 필요한 항목만 AndroidManifest.xml 파일에 추가합니다. 이 외에 안드로이드에서 정의한 모든 권한 목록을 보려면 [Manifest.permission](#)을 참조하시기 바랍니다.



기능에 필요한 권한을 설정하지 않으면 앱 실행시 해당 권한이 없다는 메시지와 함께 기능이 동작하지 않습니다.

```

<!-- AudioPlayer, AudioRecorder -->
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />

```

```

<!-- Call -->
<uses-permission android:name="android.permission.CALL_PHONE" />

```

```

<!-- Contact -->
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />

```

```

<!-- Camera -->
<uses-permission android:name="android.permission.CAMERA" />
<activity android:name="com.nexacro.deviceAPI.CameraListener" android:screenOrientation="
landscape">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>

```



```
<!-- Geolocation -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
```

```
<!-- Map -->
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-library android:name="com.google.android.maps" />
<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="user value" />
```



구글맵을 사용하려면 API KEY 값을 받아서 입력해야 합니다. 자세한 내용은 다음 링크를 참조하십시오.  
[https://developers.google.com/maps/documentation/android/start#obtain\\_a\\_google\\_maps\\_api\\_key](https://developers.google.com/maps/documentation/android/start#obtain_a_google_maps_api_key)

```
<!-- Notification -->
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<receiver
    android:name="com.nexacro.notification.NexacroNotificationReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <category android:name="com.nexacro" />
    </intent-filter>
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="com.nexacro" />
    </intent-filter>
</receiver>
<service android:name="com.nexacro.notification.NexacroNotificationService" />
```

```
<!-- SMS -->
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />

<receiver android:name="com.nexacro.deviceAPI.SmsRecv">
    <intent-filter>
```

```

        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.provider.Telephony.WAP_PUSH_RECEIVED" />
        <data android:mimeType="application/vnd.wap.mms-message" />
    </intent-filter>
</receiver>

```

```

<!-- Vibrator -->
<uses-permission android:name="android.permission.VIBRATE" />

```

```

<!-- FileDialog -->
<activity android:name="com.nexacro.deviceAPI.FileDialogActivity" android:screenOrientation="
sensor">
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

```

<!-- Bluetooth -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<service
    android:name="com.nexacro.deviceAPI.BluetoothLEAdapterService"
    android:enabled="true"
    android:exported="true">
</service>

```

```

<!-- ExternalAPI -->
<uses-permission android:name="android.permission.GET_TASKS" />

```

```

<!-- Etc -->
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />

```

## 8.2.6 APK에 넥사크로 애플리케이션 임베딩(Asset)

넥사크로 애플리케이션을 APK 파일에 임베딩해서 안드로이드 앱을 배포할 수 있습니다. 앱을 설치한 모바일 장치가 인터넷 연결이 안되는 경우에도 넥사크로 애플리케이션을 구동할 수 있습니다.

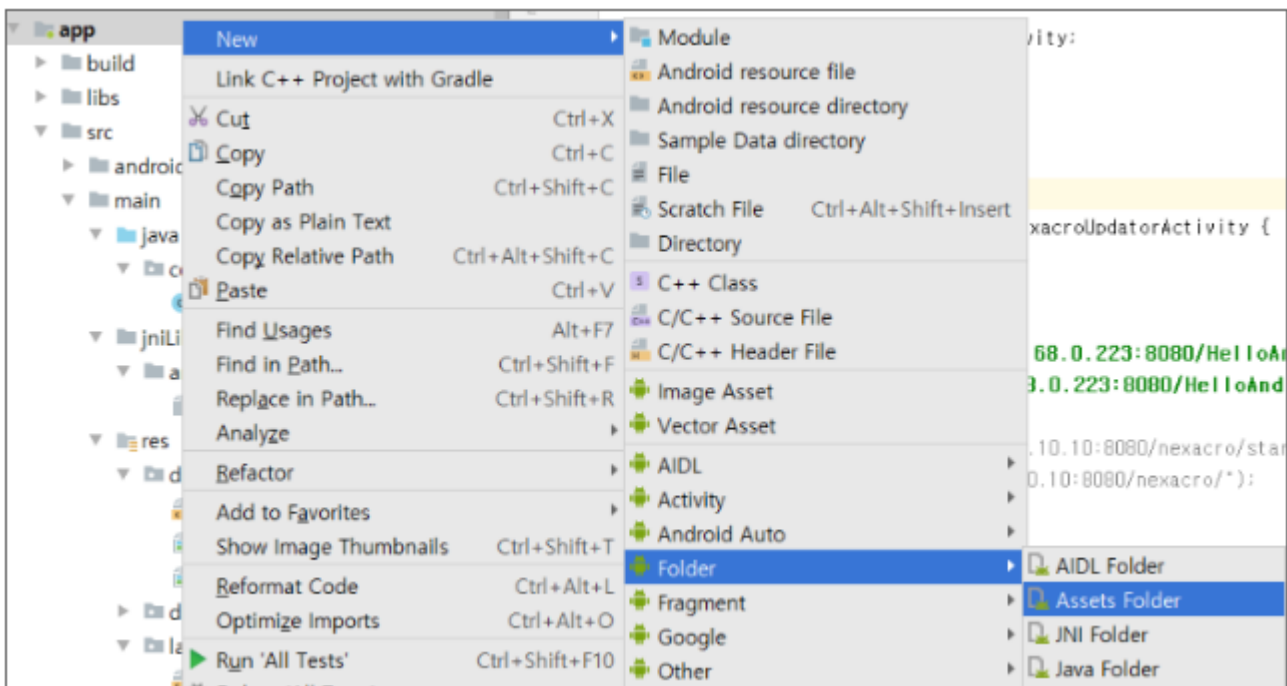
APK로의 임베딩은 넥사크로 스튜디오로 생성한 넥사크로 애플리케이션 아카이브 파일을 안드로이드 애셋(Asset)의 archive 디렉토리에 복사하는 것으로 간단히 처리할 수 있습니다.



애셋은 안드로이드 앱에서 사용할 수 있는 논리적인 저장 공간입니다. 안드로이드 프로젝트에서 assets 폴더를 만들고 앱에서 사용할 파일을 복사한 후 APK를 생성하면 앱 실행시 그 파일을 사용할 수 있습니다. 애셋은 읽기 전용 공간으로 사용되므로 앱 실행중 변경되지 않을 파일을 이 곳에 놓고 사용합니다.

### 1. Assets 폴더 생성

먼저 프로젝트의 app을 선택하고 마우스 오른쪽 버튼을 클릭합니다. 다음과 같이 컨텍스트 메뉴에서 [New > Folder > Assets Folder]를 선택하여 [프로젝트 > app > src > main] 하위에 Assets 폴더를 생성합니다.



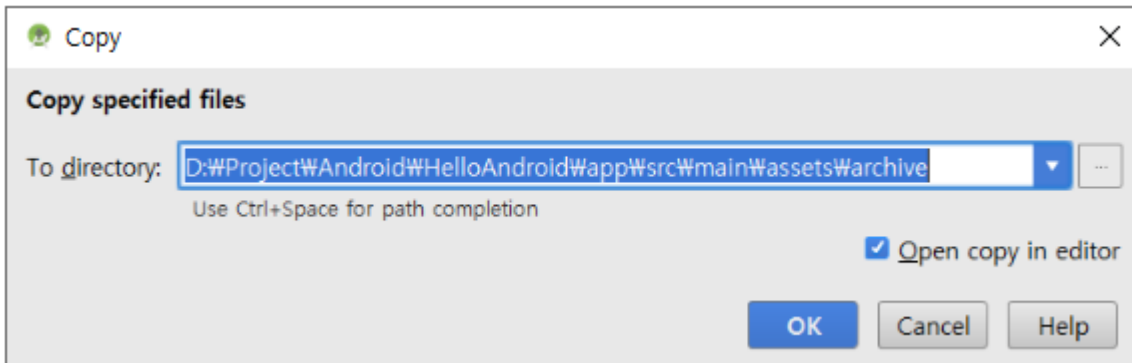
### 2. archive 디렉토리 생성

[프로젝트 > app > src > main > assets] 폴더를 선택한 후 [New > Directory] 메뉴를 사용해 archive 디렉토리를 assets 폴더 하위에 생성합니다.

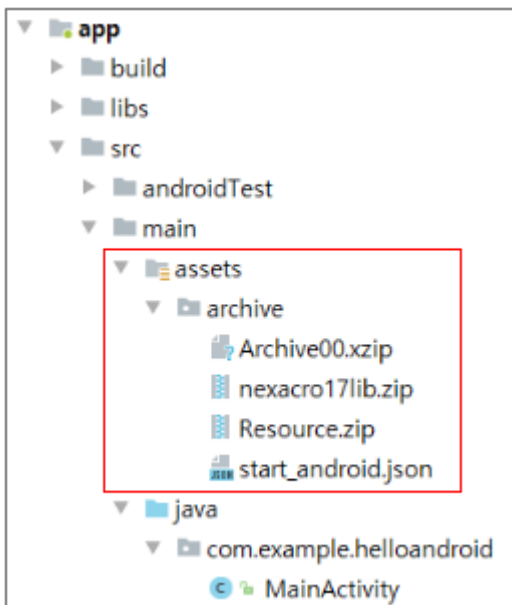
### 3. 넥사크로 애플리케이션 아카이브 파일 복사

넥사크로 애플리케이션 아카이브 파일을 [프로젝트 > app > src > main > assets > archive] 디렉토리로 복사합니다.

윈도우 탐색기에서 파일을 선택한 후 안드로이드 스튜디오로 복사/붙여넣기 합니다.



넥사크로 애플리케이션 아카이브 파일의 복사를 완료하면 다음과 같이 프로젝트 창에서 확인이 가능합니다.



## 8.3 빌드

### 8.3.1 앱 테스트

1. 안드로이드 단말기를 PC에 USB로 연결해 바로 테스트할 수 있습니다. 단말기를 직접 연결하는 경우에는 단말기에 애플리케이션 개발 옵션인 [애플리케이션 > 개발 > USB 디버깅] 항목을 설정하고 단말기에 맞는 드라이버를 설치해주어야 합니다.
2. 안드로이드 스튜디오에서 Run 메뉴를 수행합니다. Run 메뉴는 APK를 생성 및 단말기에 설치하고 실행까지 할 수 있는 기능입니다. 실행 명령을 내리면 어느 단말기에서 실행할지 선택할 수 있습니다.

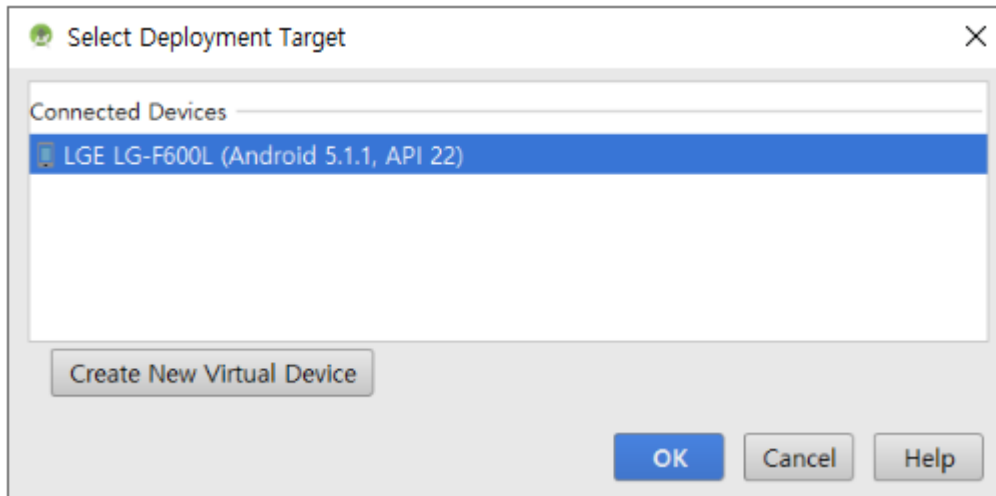
Run > Run 'App'



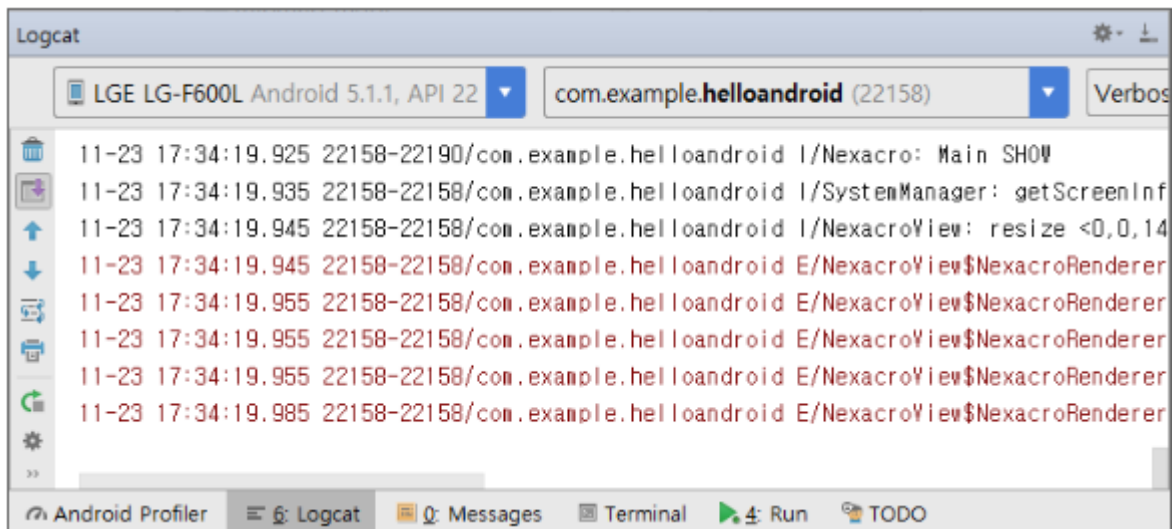
넥서스 원(Nexus One) 또는 옵티머스 원(Optimus One) 같은 경우 내장 메모리에 사용자가 쓸 수 있는 공간이 없어 외장 메모리 카드를 추가하지 않았다면 애플리케이션이 정상적으로 설치되지 않습니다.



안드로이드 가상 디바이스를 이용한 테스트는 지원하지 않습니다.



앱 실행 로그는 안드로이드 스튜디오 하단의 Logcat을 통해 확인할 수 있습니다.



## 8.3.2 설치 파일 생성

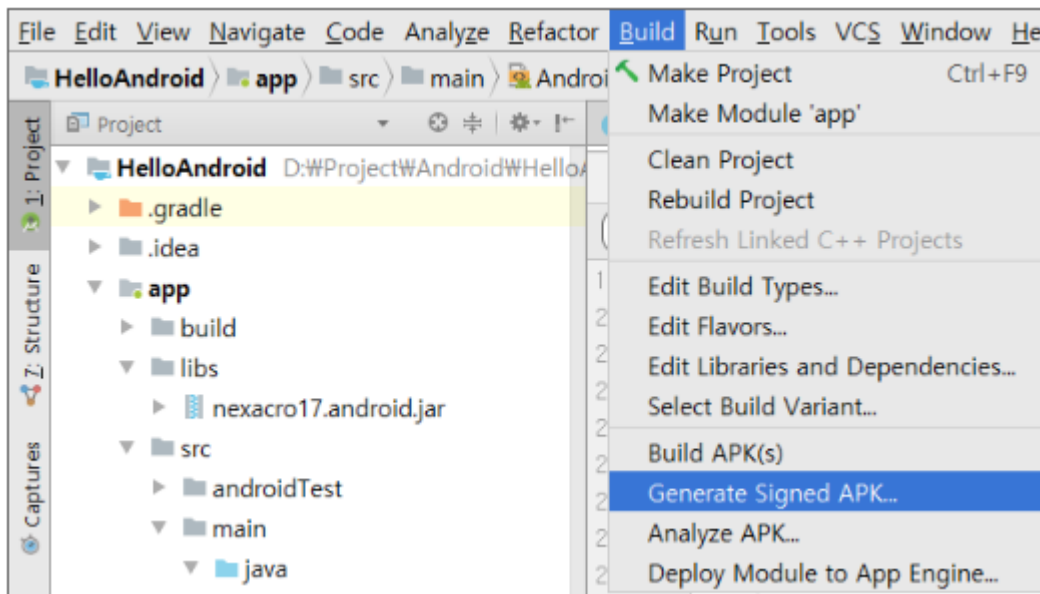
완성된 앱을 배포하기 위해서는 앱을 인식할 수 있는 keystore를 생성하고 서명된 APK 파일을 생성해야 합니다. key store 파일은 최초 1회 생성하며 이후 앱이 업데이트되었을때는 최초 생성된 keystore 파일을 사용합니다.

keystore 생성 및 Signed APK 를 생성하는 방법은 구글의 웹페이지를 참조하십시오.



<http://developer.android.com/guide/publishing/app-signing.html>

상단 메뉴에서 [Build > Generate Signed APK] 항목을 선택합니다.



## 9.

# 앱 개발 및 실행 (macOS)

넥사크로플랫폼으로 개발한 서비스를 맥 운영체제 사용자에게 하이브리드 운영방식으로 배포할 수 있습니다. 개발자가 작성한 프로젝트를 가지고 관리자는 배포 파일을 생성해야 합니다. 이번 장에서는 배포 파일을 생성하는 과정에 대한 간략한 가이드를 제공합니다.

## 9.1 준비 사항

### 9.1.1 개발 환경

맥 운영체제에서 사용할 수 있는 앱을 생성하기 위해서는 Xcode 개발 환경이 필요합니다. 추가적인 코딩 과정은 없고 이미 만들어진 넥사크로플랫폼 애플리케이션을 맥 운영체제에서 사용할 수 있도록 만드는 작업만 필요합니다.

기본적인 개발 환경은 아래와 같습니다.

항목	설명
Xcode	설치가 필요할 수 있습니다. <a href="https://developer.apple.com/xcode/">https://developer.apple.com/xcode/</a>
xcodebuild	Xcode 설치 시 같이 설치됩니다.

## 9.2 앱 프로젝트 개발

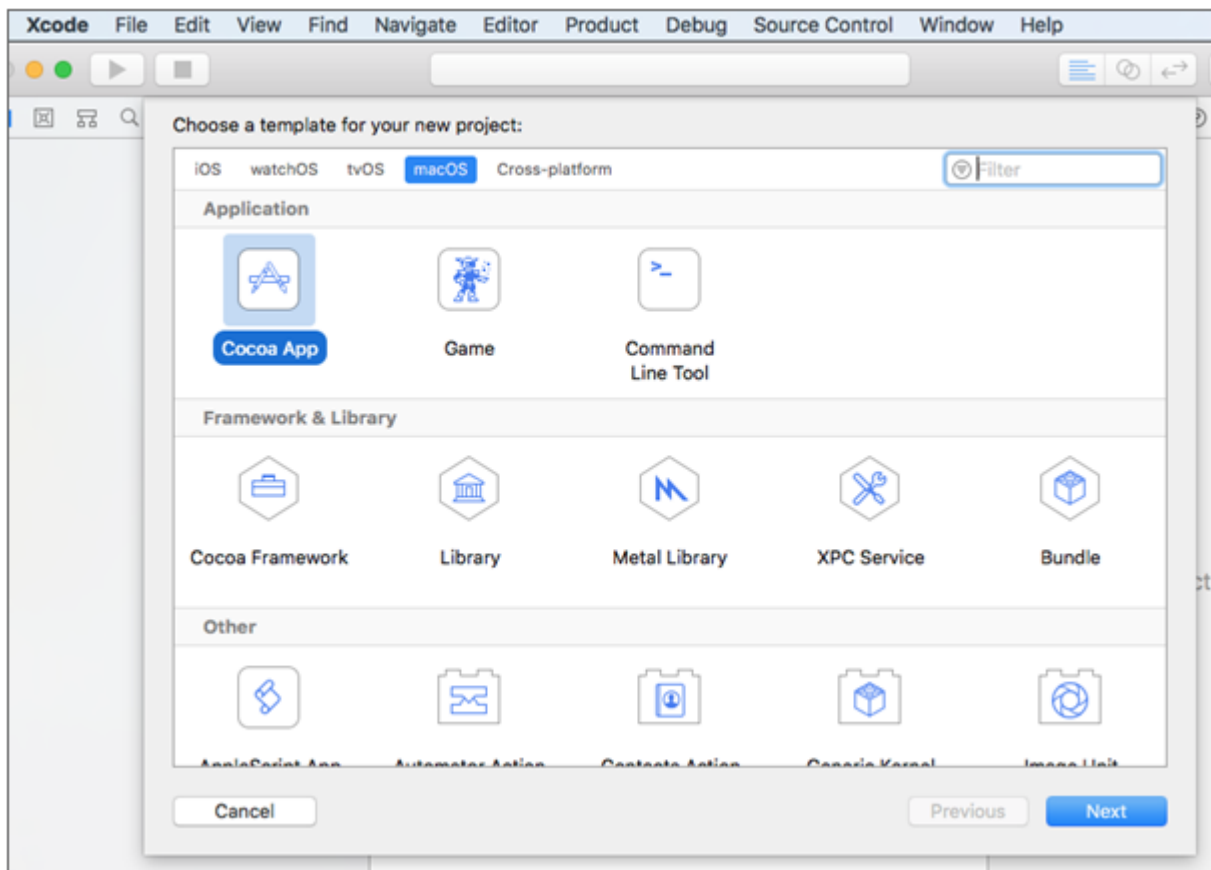
Xcode에서 아래와 같은 절차에 따라 macOS용 앱 프로젝트를 개발합니다. 앱 프로젝트를 진행하기 전에 넥사크로 스튜디오에서 개발된 애플리케이션에서 만들어진 아카이브 파일은 지정된 경로에 위치해야 합니다.

## 9.2.1 프로젝트 생성

넥사크로플랫폼으로 개발된 애플리케이션을 담을 macOS 프로젝트를 생성하고 기본 환경을 설정해야 합니다. 새로운 프로젝트는 아래 메뉴에서 생성할 수 있습니다.

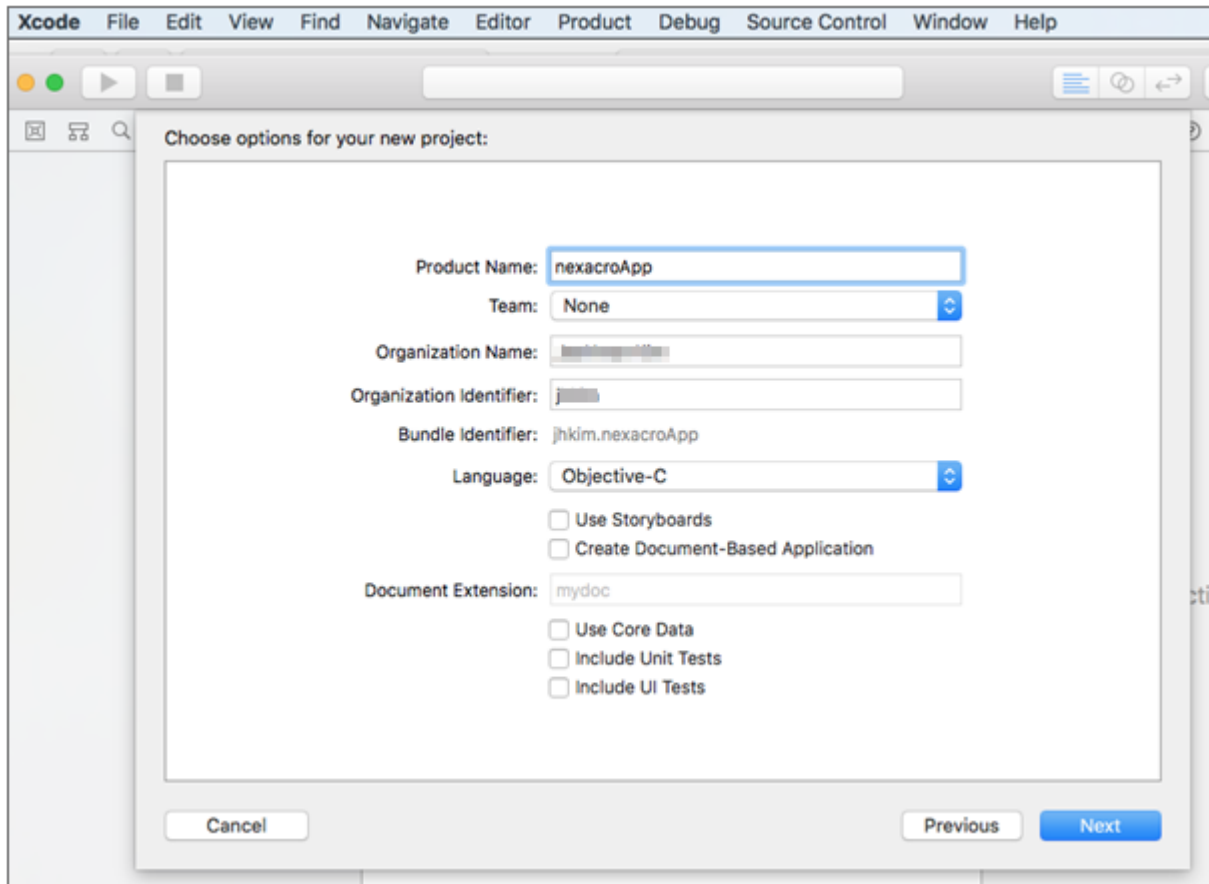
File > New > Project > macOS

프로젝트 생성을 위한 템플릿 화면에서 'Cocoa App' 항목을 선택합니다.



Language는 "Objective-C"로 설정하고 Product Name과 기타 필요한 정보를 입력하고 프로젝트를 생성할 폴더 위치를 지정한 후 [Create] 버튼을 클릭합니다.



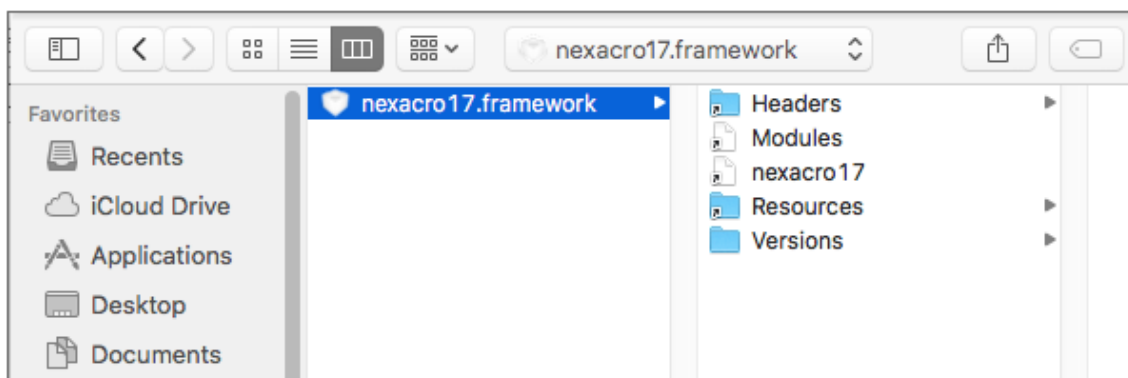


## 9.2.2 넥사크로플랫폼 라이브러리 설정

macOS 프로젝트에서 넥사크로플랫폼에 최적화된 환경을 만들기 위해 추가로 제공되는 넥사크로플랫폼 라이브러리 파일을 설정합니다.

넥사크로플랫폼 라이브러리는 압축 파일 형태로 제공되며 nexacro17.macOS.framework.zip이라는 파일명으로 제공됩니다. 제공되는 파일은 압축을 풀어 생성된 프로젝트의 임의의 폴더에 끌어다 놓거나 [Add Files to] 메뉴를 통해 추가합니다.

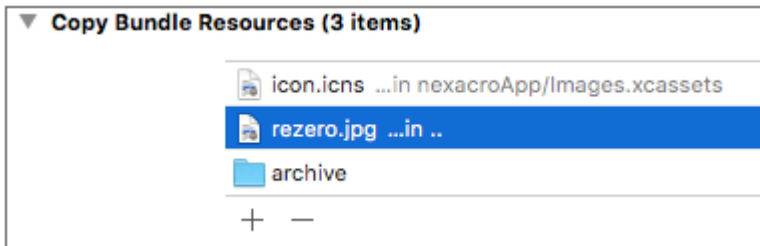
File > Add Files to "..."



## 9.2.3 리소스 설정

앱에서 사용할 로딩 이미지, 아이콘, 메시지, 레이아웃 등을 설정하는 단계입니다. 진행하는 프로젝트에 따라 변경해 적용할 수 있습니다.

Copy Bundle Resources 에 이미지 파일 추가 후 main.m 파일을 수정합니다.



```
startManager.splashImageName = @"rezero.jpg";
```

## 9.2.4 빌드 환경 설정



기본으로 만들어진 파일 중에 사용하지 않는 AppDelegate.h, AppDelegate.m, MainMenu.xib 파일은 삭제합니다.

### main.m

Supporting Files 폴더 아래에 있는 main.m 파일을 아래와 같이 수정합니다.

```
// main.m

#import <Cocoa/Cocoa.h>
#import "nexacro17/nexacro.h"

int main(int argc, const char * argv[]) {
    NexacroStartManager *startManager = [NexacroStartManager sharedManager];

    // 생략시 애플리케이션 이름으로 자동 설정
    startManager.applicationKey = @"";

    // 필수 입력값이며 디폴트 부트스트랩 파일명을 사용할 경우 start_macos.json 파일명은
```

### 생략가능

```
startManager.bootstrapUrl = @"http://172.10.12.180:8080/Mobile_Test/start_macos.json";

// 생략시 bootstrap파일이 있는 경로로 자동 설정
startManager.projectUrl = @"http://172.10.12.180:8080/Mobile_Test/";

// 초기 로딩화면 설정 (리소스 폴더를 참조함. 생략시 넥사크로 디폴트 로딩 화면을 사용함)
// startManager.splashImageName = @"sample.png";

[NSApplication sharedApplication];
[NSBundle loadNibFile:[startManager getNextacroNibFile] externalNameTable:nil withZone:nil];
[NSApp run];

return EXIT_SUCCESS;
}
```

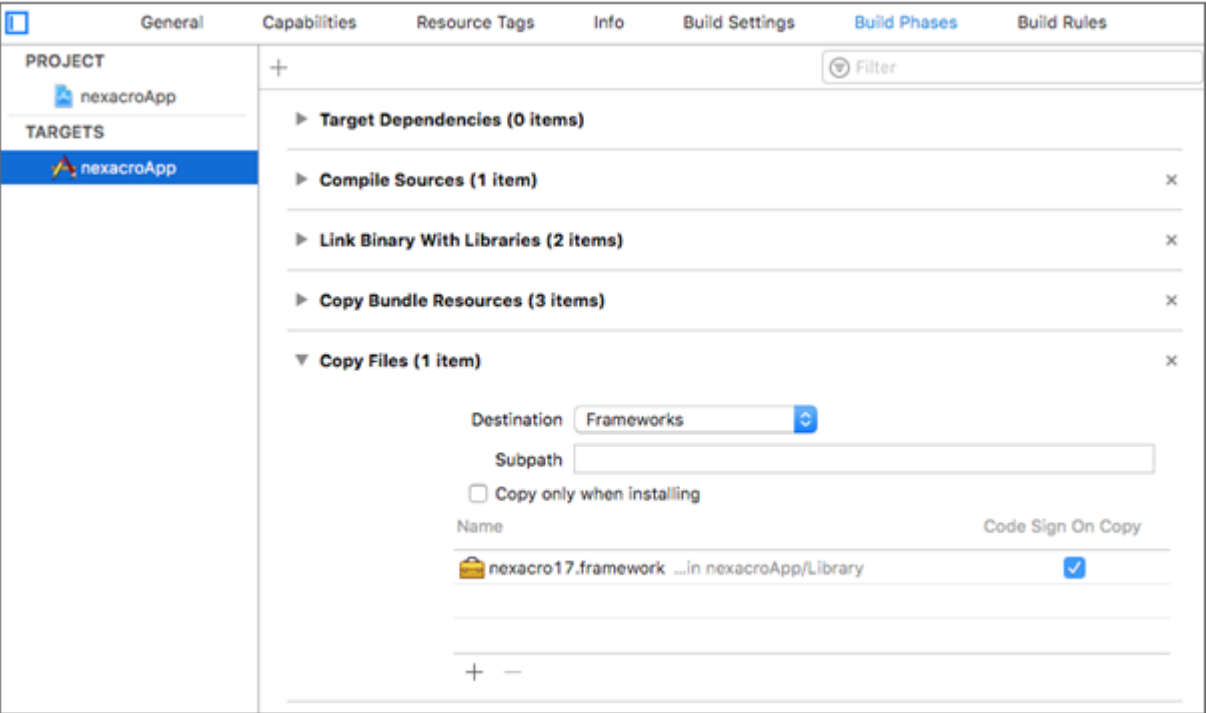
## 기타 설정

General	Deployment Target: 10.7 Main Interface: MainMenu를 삭제
Build Settings	Architectures: Universal(32/64-bit intel) - \$(ARCHS_STANDARD_32_64_BIT) Valid Architectures: i386 x86_64 macOS Deployment Target: macOS 10.7 Objective-C Automatic Reference Counting: Yes




2018년 출시될 macOS부터 32비트 앱 지원은 deprecated로 변경되고, 2019년부터는 iOS 11처럼 32비트 앱으로 만들어진 앱은 실행이 되지 않습니다.

- Build Phases - Copy Files' 항목에서 프레임워크 추가  
(+) 클릭 후 > New Copy Files Phase > Destination: Frameworks  
nexacro17.framework 추가



- info 항목에서 아래 설정 추가 또는 변경  
App Transport Security Settings > Allow Arbitrary Loads

▼ App Transport Security Settings	▲ ▼	Dictionary	(1 item)
Allow Arbitrary Loads	▲ ▼	Boolean	YES



Allow Arbitrary Loads 항목값을 "YES"로 설정하지 않으면 WebBrowser 컴포넌트가 화면에 표시되지 않습니다.

## 부록 A.

# 경로설정

## A.1 Alias 경로

넥사크로플랫폼은 사용자 PC의 경로설정 시 다음의 Alias 경로를 사용합니다.

Alias 경로 명	실제 경로명
%NEXACRO%	%PROGRAMFILES%₩nexacro₩17
%NEXACRO RESOURCE%	%PROGRAMFILES%₩nexacro₩17
%USERAPP%	%LOCALAPPDATA%LOW₩nexacro₩17 ₩data₩data₩[packageName]₩files₩nexacro₩ (Android) ₩Library₩Caches₩nexacro₩ (iOS)
%UPDATE%	%USERAPP%₩Update
%CACHE%	%USERAPP%₩Cache
%WEBDEPLOY SETUP%	Server Context root
%WEBDEPLOY PROJECT%	Server Context root
%WEBDEPLOY FRAMEWORK%	%WEBDEPLOY PROJECT%₩nexacro17lib₩framework
%WEBDEPLOY COMPONENT%	%WEBDEPLOY PROJECT%₩nexacro17lib₩component
%WEBDEPLOY RESOURCE%	%WEBDEPLOY PROJECT%₩nexacro17lib₩resources
%WEBDEPLOY THEME%	%WEBDEPLOY PROJECT%₩_resource_₩_theme_

이 Alias 경로는 배포정보의 PATH나 응용프로그램에서 PATH 정보를 입력할 때, 사용할 수 있습니다.



설치 파일 배포 시 Setup Manager를 사용해 설치 경로를 수정한 경우에는 %NEXACRO% 실제 경로가 달라질 수 있습니다.

## A.2 nexacro.xml

nexacro.xml 파일은 사용자별로 정보를 담기 위해 사용하는 XML 파일입니다. 파일 경로는 아래와 같습니다.

%USERAPP%

## A.3 상대경로

### A.3.1 Project 내 경로의 상대경로 지원여부

아래 표는 Project 내에서 주요경로의 종류 및 상대경로 지원여부, Service이용경로(Service::a.xfdl형태의 경로)의 지원여부를 정리한 것입니다.

File종류	항목	상대경로		Service이용경로 지원여부
		지원여부	기준경로	
Typedefinition	Component UpdateURL ❶	지원	TypeDefintion	지원
	Service 경로	지원	TypeDefintion	지원 불가능
GlobalVariable	Image	지원	GlobalVariable	지원
XADL	TypeDefinition ❷	지원	XADL	지원 불가능
	GlobalVariable	지원	XADL	지원
	EngineUrl ❸	지원	XADL	지원 불가능
	LiceneceUrl ❹	지원	XADL	지원 불가능
	Themeid	지원	XADL	지원

#### 1. Component UpdateURL

Component를 Update하기 위한 서버 경로를 말하며 Tool에서 TypeDefinition Update Url을 의미합니다.

단, 이 경로의 Domain(예:http://a.b.c/path에서 a.b.c)이 XADL의 경로와 일치하지 않으면 Update 하지 않습니다. 만일, XADL이 Local 경로일 경우는 Domain정보에 상관없이 Update합니다.

#### 2. TypeDefinition

Service의 경로를 말하며 Tool에서 TypeDefinition Services url을 의미합니다.

#### 3. Engine 경로

Engine 업데이트를 위한 경로를 말하며 Tool의 ADL engineurl을 의미합니다. 단 XADL경로와 Domain이 일치하지 않으면 Update 하지 않습니다. XADL이 Local 경로일 때는 Domain에 상관없이 Update합니다.

#### 4. 라이선스 경로

Typedefinition을 읽기 전에 라이선스를 체크하기 위한 경로입니다.

Tool의 ADL licenseurl을 의미합니다.

단, XADL경로와 Domain이 일치해야 합니다. XADL이 Local경로일 때는 Domain에 상관없이 라이선스 파일을 체크합니다.

## 부록 B.

### 예외상황

운영 과정에서 예외적으로 발생할 수 있는 상황에 대한 대처 방안을 정리한 것입니다. 사용 환경에 따라 적용 방식이 달라질 수 있으니 참고로만 활용하시기 바랍니다.

#### B.1 웹브라우저 옵션

사용자가 웹브라우저에서 특정한 목적으로 옵션을 변경했을 경우에 애플리케이션에서 해당 기능을 사용하고 있다면 정상적인 동작을 하지 못할 수 있습니다. 주로 아래와 같은 상황에서 문제가 생길 수 있습니다.

##### B.1.1 자바스크립트 활성화

사용자가 웹브라우저에서 자바스크립트를 사용할 수 없도록 설정했을 경우에는 애플리케이션이 정상적으로 동작하지 않습니다. 일반적으로 기본 옵션은 활성화 상태이지만 사용자가 다른 옵션을 선택할 수 있으므로 애플리케이션이 화면에 보이지 않을 때는 해당 옵션을 먼저 확인합니다.

사용하는 웹브라우저에 따라 자바스크립트 사용 옵션을 활성화해주어야 합니다. 사용하는 버전에 따라 해당 옵션은 달라질 수 있으며 상세한 내용은 아래 링크 또는 각 웹브라우저 도움말을 참고하세요.

<http://www.enable-javascript.com>

인터넷 익스플로러의 경우에는 아래 옵션에서 변경합니다.

Tools > Internet options > Security > Custom level > Internet Zone > Scripting > Active scripting



## B.1.2 파일 다운로드 활성화

사용자가 웹브라우저에서 파일 다운로드를 사용할 수 없게 제한한 경우에는 Filddownload 컴포넌트를 사용해 파일을 내려받을 수 없습니다. 일반적으로 기본 옵션은 활성화 상태이지만 사용자가 다른 옵션을 선택할 수 있으므로 파일 다운로드가 동작하지 않을 때는 해당 옵션을 먼저 확인합니다.

사용하는 웹브라우저에 따라 파일 다운로드 옵션을 활성화해주어야 합니다. 사용하는 버전에 따라 해당 옵션은 달라질 수 있으며 상세한 내용은 각 웹브라우저 도움말을 참고하세요.

인터넷 익스플로러의 경우에는 아래 옵션에서 변경합니다.

Tools > Internet options > Security > Custom level > Internet Zone > Downloads > File Download

## B.1.3 HTTP 1.1 활성화

사용자가 웹브라우저에서 HTTP 1.1을 사용할 수 없게 제한한 경우에 애플리케이션이 느려지거나 동작하지 않을 수 있습니다. 일반적으로 기본 옵션은 활성화 상태이지만 사용자가 다른 옵션을 선택할 수 있으므로 애플리케이션이 동작하지 않을 때는 해당 옵션을 먼저 확인합니다.

인터넷 익스플로러에서만 제공하는 기능이며 아래 옵션에서 변경합니다.

Tools > Internet options > Advanced > HTTP settings > Use HTTP 1.1 through proxy connections

Tools > Internet options > Advanced > HTTP settings > Use HTTP 1.1



웹브라우저 설정과 상관없이 웹서버 설정에서 HTTP 1.0을 사용하도록 강제하는 경우에 애플리케이션이 느려지거나 동작하지 않을 수 있습니다. 전체 사용자 환경이 느려지는 경우에는 웹서버 설정을 확인하셔야 합니다.

아파치 서버의 경우에 force-response-1.0 설정을 사용할 수 있습니다.

<http://httpd.apache.org/docs/2.2/en/env.html#special>

## B.1.4 XMLHTTP 활성화

사용자가 웹브라우저에서 XMLHTTP를 사용할 수 없게 제한한 경우에 애플리케이션 내에서 동적인 처리 작업을 수행하지 못할 수 있습니다. 일반적으로 기본 옵션은 활성화 상태이지만 사용자가 다른 옵션을 선택할 수 있으므로 애플리케이션이 동작하지 않을 때는 해당 옵션을 먼저 확인합니다.

인터넷 익스플로러에서만 제공하는 기능이며 아래 옵션에서 변경합니다.

Tools > Internet options > Advanced > Security > Enable native XMLHTTP support

## B.2 인터넷 익스플로러 호환성 보기

사용자가 인터넷 익스플로러에서 서비스되고 있는 도메인을 호환성 보기 대상으로 추가한 경우에는 화면이 정상적으로 출력되지 않을 수 있습니다.

넥사크로플랫폼 애플리케이션 개발 시 사용하는 HTML 파일은 상위 버전 인터넷 익스플로러의 향상된 성능을 이용하기 위해 각 버전의 표준모드로 동작하도록 Meta tag로 렌더링 모드를 지정하였으며 (IE=Edge), 해당 모드에서는 주소 입력창에 호환성 보기 아이콘이 표시되지 않습니다.

따라서 화면이 깨지는 사용자의 브라우저 설정을 확인하여 표준 모드로 동작하도록 가이드를 제공하거나, 별도의 스크립트로 호환성 보기 모드인지를 확인하여 사용자에게 알려 줄 수 있습니다.



아래 코드는 참고용입니다. 개발 환경에 따라 사용하셔야 합니다.

참고: [http://msdn.microsoft.com/en-us/library/ms537503\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms537503(v=vs.85).aspx)

```
var agentStr = navigator.userAgent;

var mode;

if(agentStr.indexOf("rv:11.0") >= 0) {
    mode = "IE11";
}
else if (agentStr.indexOf("Trident/6.0") > -1) {
    if (agentStr.indexOf("MSIE 7.0") > -1) {
        mode = "IE10 Compatibility View";
    } else {
        mode = "IE10";
    }
}
else if (agentStr.indexOf("Trident/5.0") > -1) {
    if (agentStr.indexOf("MSIE 7.0") > -1) {
        mode = "IE9 Compatibility View";
    } else {
        mode = "IE9";
    }
}
```

```

}
else if (agentStr.indexOf("Trident/4.0") > -1) {
    if (agentStr.indexOf("MSIE 7.0") > -1) {
        mode = "IE8 Compatibility View";
    } else {
        mode = "IE8";
    }
}
else {
    mode = "IE7";
}

document.title = "Browser Mode:\t" + mode;

```

## B.3 기존 웹 화면에 아이프레임으로 콘텐츠 추가

이미 개발되어 운영하는 화면에 아이프레임 형식으로 넥사크로플랫폼 콘텐츠를 추가하는 경우 인터넷 익스플로러에서 화면이 출력되지 않을 수 있습니다.

넥사크로플랫폼 애플리케이션 개발 시 사용하는 HTML 파일은 Edge모드로 동작하게 구성되어 있습니다. 기존 화면에 사용된 HTML 파일에 DTD(Document Type Definition)이 지정되어 있지 않을 경우에는 쿼크모드(Quirks mode)로 동작하는데 이 과정에서 화면이 정상적으로 출력되지 않을 수 있습니다.

이런 경우에는 아래와 같이 메타 태그를 추가해줍니다.

```

<head>
  <meta http-equiv="X-UA-Compatible" content="IE=Edge"/>
</head>

```



메타 태그 또는 HTTP 헤더를 사용한 호환성 보장과 관련된 내용은 아래 링크를 참고하세요.

<http://technet.microsoft.com/ko-kr/library/gg699448.aspx>