

# Addressable Asset System

V1.19.17

작성자 : 김은규

# 01. Setting

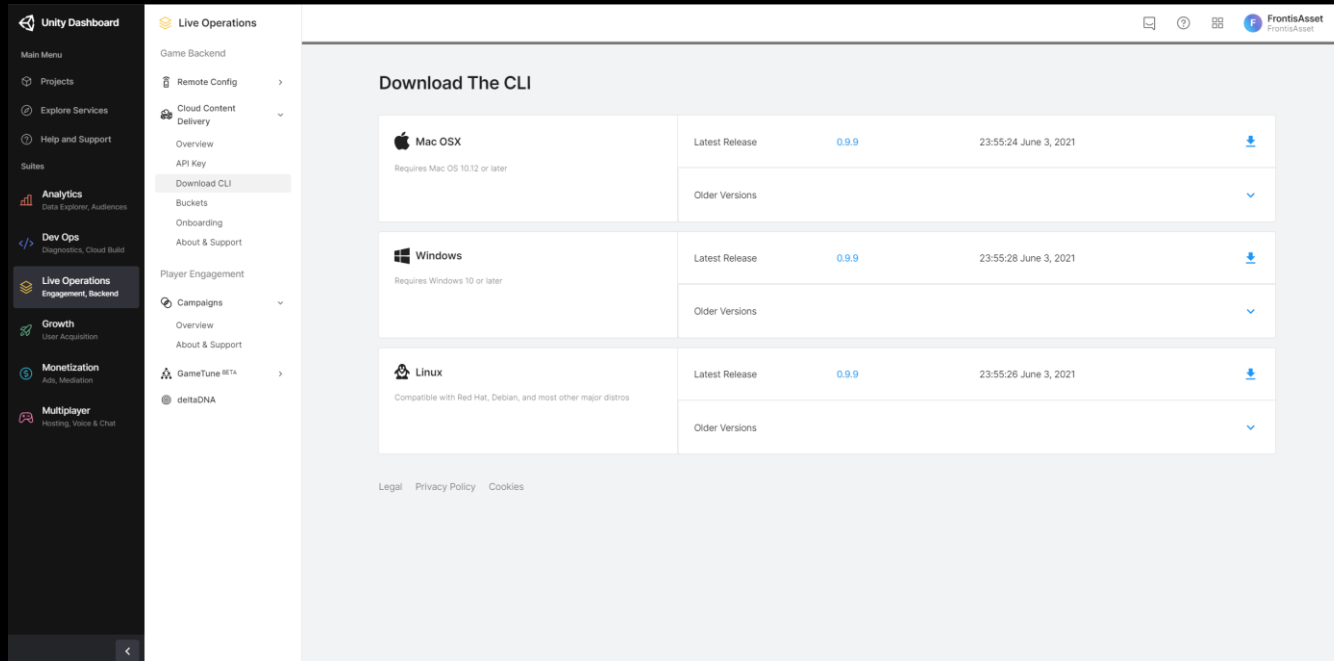
- Unity 2020.3.25f1
- Addressable System v1.19.17

## 02. Changelog

- [1.19.17] – 2022 – 1 – 6

## 02. CCD내 CLI

- CCD를 CLI를 설치하여 자동으로 에셋을 업로드 가능 합니다.

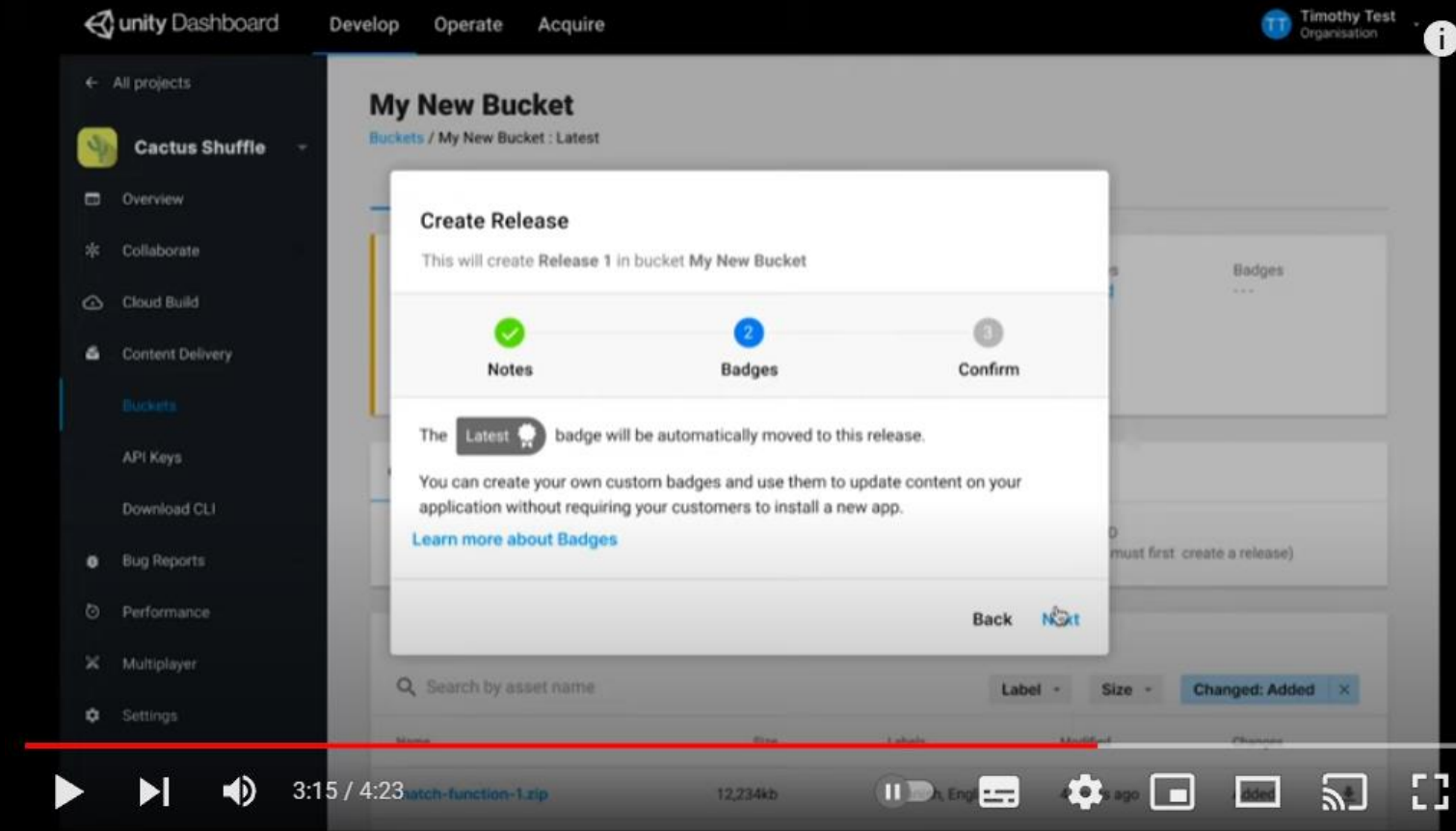


The screenshot displays the Unity Dashboard interface. On the left, a sidebar menu lists various services under 'Main Menu' and 'Suites'. The 'Live Operations' section is highlighted. The main content area is titled 'Download The CLI' and features three rows for different operating systems: Mac OS X, Windows, and Linux. Each row includes a 'Latest Release' link with the version number '0.9.9', a timestamp '23:55:24 June 3, 2021', and a download icon. Below the 'Latest Release' link is an 'Older Versions' link with a dropdown arrow. The footer of the page contains links for 'Legal', 'Privacy Policy', and 'Cookies'.

Operating System	Latest Release	Version	Timestamp	Action
Mac OS X	Requires Mac OS 10.12 or later	0.9.9	23:55:24 June 3, 2021	Download
		Older Versions		▼
Windows	Requires Windows 10 or later	0.9.9	23:55:28 June 3, 2021	Download
		Older Versions		▼
Linux	Compatible with Red Hat, Debian, and most other major distros	0.9.9	23:55:26 June 3, 2021	Download
		Older Versions		▼

# 03. Get started with Cloud Content Delivery

<https://www.youtube.com/watch?v=-RhtMfpqILY>



# 04. Work Flow

## 주소 지정 가능 워크플로

자산(예: Prefab)이 "주소 지정 가능"으로 표시되면 어디에서나 호출할 수 있는 주소가 생성됩니다. 자산이 어디에 있든(로컬 또는 원격) 시스템은 자산과 해당 종속성을 찾은 다음 반환합니다. 자산은 게임이나 앱을 만드는 데 사용하는 콘텐츠입니다. 자산의 일반적인 예로는 프리팹, 텍스처, 재료, 오디오 클립 및 애니메이션이 있습니다.

Addressables는 자산 번들을 추상화하여 콘텐츠 관리를 보다 효율적으로 만드는 동시에 자산과 모든 기본 데이터를 포함합니다.

주소 지정 워크플로에서 요청은 먼저 카탈로그 시스템을 통과합니다. 시스템은 자산이 무엇인지, 자산의 종속성, 로컬인지 원격인지 등에 대해 빌드 중에 생성된 데이터로 구성된 위치로 주소를 디코딩합니다.

카탈로그에서 요청은 공급자 시스템을 통해 전달됩니다. 하나 이상의 공급자가 이러한 위치를 사용하여 콘텐츠를 찾은 다음 장치에 반환합니다.

Addressables에서 런타임은 비동기식입니다. 이를 통해 게임 코드를 변경하지 않고도 자산이 어디에 있든 필요할 때 자산을 검색할 수 있는 유연성을 제공합니다. 해당 위치는 현재 개발 단계에 따라 변경될 수 있습니다.

# 05. Addressables Groups

## 주소 지정 가능 그룹

어떤 것을 주소 지정 가능으로 표시하면 주소 지정 가능 자산 및 해당 데이터의 컨테이너인 주소 지정 가능 그룹의 일부가 됩니다. Addressables의 홈 베이스인 Groups 창에서 그룹을 시각화할 수 있습니다. 그룹은 해당 자산이 장치에 로컬인지 아니면 서버에 있는지 여부를 결정할 수 있습니다.

Addressables Groups의 데이터는 데이터 계약인 스키마에 보관됩니다. 스키마 중 하나는 자산과 콘텐츠가 번들로 구축되는 방식에 중점을 둡니다. 빌드 경로 및 로드 경로는 드롭다운 메뉴에서 변수를 선택하여 콘텐츠를 로컬 또는 원격으로 설정할 수 있는 보다 유용한 설정입니다.

주소 지정 가능 프로필 시스템에서 주소 지정 가능 그룹에 대한 프로필을 만든 다음 지정된 프로필에 대해 변수를 평가할 대상을 정의합니다. 이를 통해 그룹의 데이터를 설정하고 해당 그룹의 일부를 다시 코딩하지 않고도 원격으로 변경할 수 있습니다.

## o6. Create

### 생산

어드레스블을 최대한 활용하려면 게임을 배송하려는 방식과 관련하여 데이터를 시각화하는 데 도움이 됩니다. 개발 중에 Profile 변수를 변경하기 쉽기 때문에 구조에 커밋할 필요가 없지만 번들 구성에 대한 일반적인 접근 방식이 있어야 합니다.

예를 들어 게임을 원격 콘텐츠와 함께 제공하지만 개발 중에 해당 콘텐츠를 로컬로 유지하려는 경우 원격 경로가 스트리밍 자산을 가리키는 프로필을 만들 수 있습니다. 이런 식으로 그룹의 코드를 건드리지 않고도 모든 원격 콘텐츠를 로컬로 변경할 수 있습니다.



## 07. Editor Hosted 구조

### Editor Hosted

Profile Name	LocalLoadPath	RemoteLoadPath
Production	{UnityEngine.AddressableAssets.Addressables.RuntimePath}	http://myinternet.com/[BuildTarget]
Local	{UnityEngine.AddressableAssets.Addressables.RuntimePath}	{UnityEngine.AddressableAssets.Addressables.RuntimePath}
Hosted Prod	{UnityEngine.AddressableAssets.Addressables.RuntimePath}	http://{PrivateIpAddress_2}:{HostingServicePort}

#### Local Content In Player

##### Player Data

DLLS  
Resources  
Scenes

##### Local Bundles

#### Hosted on my computer

##### Remote Bundles

# o8. Editor Hosting 설명

## 에디터 호스팅

때로는 서버에서 콘텐츠를 호스팅해야 합니다. 원격 경로를 설정하고 실제 URL을 가리키는 대신 호스팅 서비스에서 정의한 변수를 사용할 수 있습니다.

호스팅을 활성화하면 호스팅 서비스가 에디터 내에서 HTTP 호스트를 설정합니다. 장치나 플레이어를 이 호스트에 연결하여 테스트할 수 있습니다.

에디터 호스팅의 강력한 기능 중 하나는 모든 것을 원격으로 설정할 수 있다는 것입니다. 이는 콘텐츠 개발자와 아티스트에게 특히 유용합니다. 콘텐츠를 계속 반복하면서 플레이어를 빌드하고 기기에 배포할 수 있기 때문입니다. 플레이어를 재배포하거나 한 장치에서 다른 장치로 콘텐츠를 이동하는 것에 대해 걱정할 필요가 없습니다. 주소 지정 가능으로 호스팅 서비스를 만들고 구성하는 방법에 대한 자세한 내용은 [설명서](#) 를 확인하십시오 .

Addressables로 해결된 동적 콘텐츠 지원의 주요 기술 과제 중 상당수가 라이브 프로덕션 게임 및 앱에 대한 자산의 호스팅 및 전달이라는 "라스트 마일" 문제가 여전히 남아 있습니다. 올해 말, 우리는 Addressables 시스템에 완전히 통합된 엔터프라이즈급 글로벌 콘텐츠 호스팅 솔루션을 출시할 것입니다. 이 서비스에 대해 자세히 알아보려면 [여기](#) 에서 [등록하십시오](#) .

# 09. Addressables Asset Load

## Addressables에서 자산을 로드하는 방법

라이브 콘텐츠 팀은 자산을 로드하는 몇 가지 빠른 방법을 포함하여 작업 속도를 높이기 위한 워크플로를 작업하고 있습니다.

**Addressables에서 자산을 어떻게 로드할 수 있습니까?**

**주소별:** 코더는 쉽게 런타임 검색을 위해 자산의 위치 식별자를 사용하여 문자열로 주소 지정 가능 항목을 로드하는 경우가 많습니다.

**레이블별:** 유사한 항목의 런타임 로드를 위한 추가 주소 지정 가능 자산 식별자를 제공합니다.

**AssetReference 기준:** AssetReference는 직접 참조처럼 작동하지만 초기화가 지연됩니다.

AssetReference 개체는 GUID를 요청 시 로드할 수 있는 주소 지정 가능으로 저장합니다. [에디터에서 작업하는](#) 아티스트는 이 워크플로를 선호할 수 있습니다.

참조하는 자산에 하위 개체(예: SpriteAtlas 내의 스프라이트)가 있는 경우 하위 개체를 추가로 참조할 수 있습니다.

Addressables와 스프라이트 로딩 과정을 보고 싶다면 세션의 [스프라이트 데모 부분](#)을 확인하세요 .

# 10. 빌드 사용자 지정

## 빌드 사용자 지정

Addressables 패키지에는 앱 개발을 가속화하는 데 도움이 되는 재생 모드 데이터를 생성하는 세 가지 빌드 스크립트가 있습니다. 스크립트는 *자산 데이터베이스 사용*, *그룹 시뮬레이션* 및 *기존 빌드 사용*입니다.

*사용 자산 데이터베이스* 스크립트. 이를 통해 콘텐츠를 반복하는 동안 재생 모드 안팎으로 이동할 수 있습니다. 게임의 흐름을 따라 작업하면서 게임을 빠르게 실행할 수 있습니다. 분석이나 자산 번들 생성 없이 빠른 반복을 위해 자산 데이터베이스를 통해 자산을 직접 로드합니다.

*Simulate Groups* 는 자산 번들을 생성하지 않고 레이아웃 및 종속성에 대한 콘텐츠를 분석합니다. 게임 플레이 중 번들이 로드 또는 언로드되는 시기를 보려면 주소 지정 가능한 이벤트 뷰어 창(창 > 자산 관리 > 주소 지정 가능 > 이벤트 뷰어)에서 자산 사용량을 확인하십시오. 이 모드는 로드 전략을 시뮬레이션하고 콘텐츠 그룹을 조정하여 프로덕션 릴리스에 적합한 균형을 찾는 데 도움이 됩니다.

*사용하는 기존 빌드* 스크립트는 배포된 응용 프로그램 빌드와 비슷하지만 별도의 단계로 데이터를 구축할 수 있어야 합니다. 자산을 수정하지 않는 경우 이 모드는 재생 모드에 들어갈 때 데이터를 처리하지 않으므로 가장 빠릅니다.

# Use Asset Database(fase)

- Catalog path :
- Project Folder
- /Libray/com.unityaddressables/catalog\_BuildScriptFastMode.json
- Settings path :
- /settings\_BuildScriptFastMode.json
- 번들~~을~~ 빌드하지 않고, Entry Point를 통해 직접 접근하므로 Load가 빠르다.

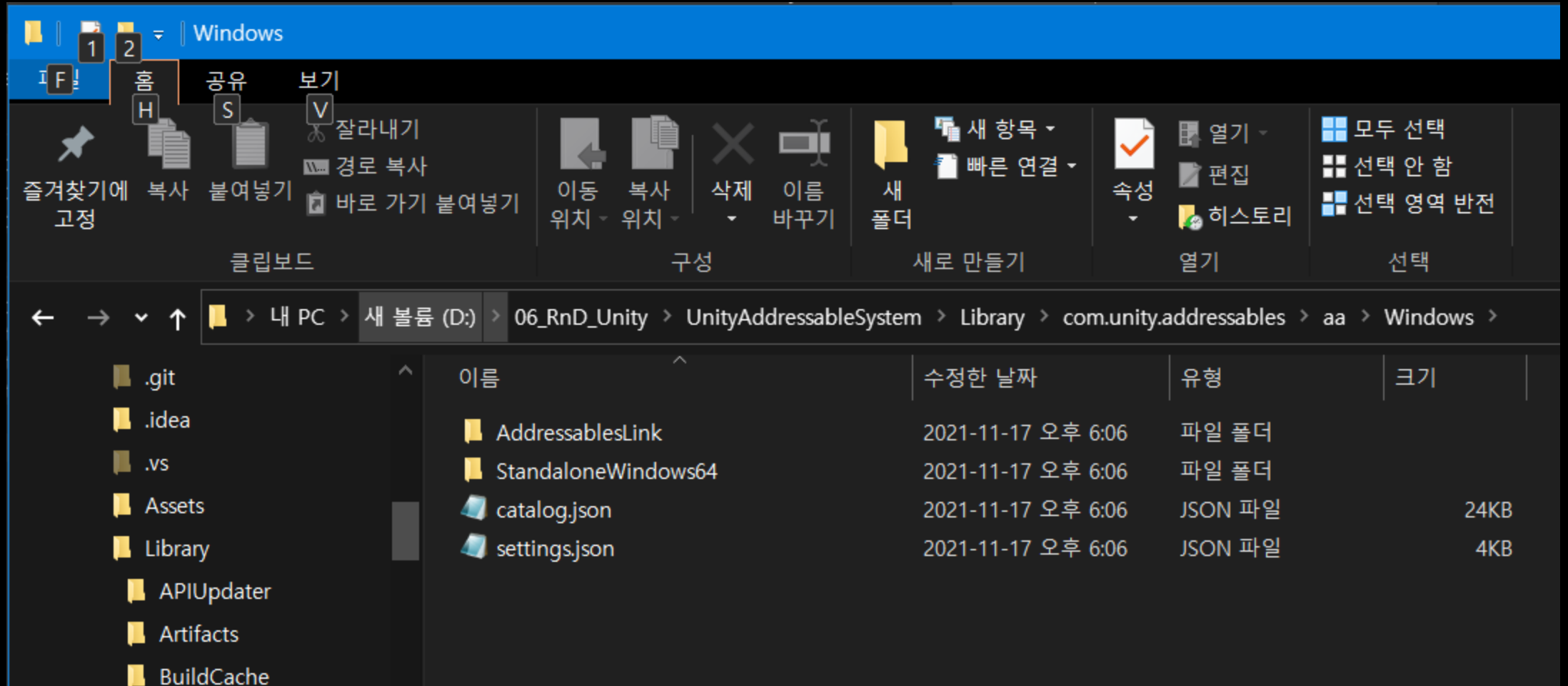
# Simulate Group(advanced)

- Catalog path :
- Project Folder
- /Libray/com.unityaddressables/catalog\_BuildScriptVirtualMode.json
- Settings path :
- /settings\_BuildScriptVirtualMode.json
- 가상의 bundle을 빌드해서 그 bundle을 통해 Asset에 접근하는 방식으로, bundle간 dependency가 제대로 되어있는지, 모바일 빌드했을 때 참조에러가 나지 않는지 editor 환경에서 bundle 참조확인이 가능한 모드

# Use Existing Build (requires built groups)

- 스크립트를 통해 빌드를 이미 수행했다는 전제하에, 물리적으로는 존재하는 bundle을 참조해서 Asset에 접근하는 방식
- 개발 중 직접 bundle을 빌드해서 확인하고자 할때 사용
- Catalog와 settings 파일은 빌드시 에
- Proecjt  
Foloder/Library/com.unity.addressables/StreamingAssetCop/aa/[BuildTarget]에 만들어지므로 참조 경로도 설정

# 카탈로그 세팅 파일





# 세팅 파일

## Settings.json

- 이 프로젝트가 참조해야 할 catalo파일이 어디 존재하는지 명시
- Hash파일과 remote hash파일이 어디 존재하는 지 명시
- 기타 Addressables 관련 환경변수가 저장되어 있음

# 카탈로그

## Catalog.json

- 실질적인 bundl들의 정보로서 모든 addressables의 이름, 경로, Key, ResourceType이 정의 되어 있음
- 해당 파일을 이용하면 이전 catalog와의 diff를 통해 자체적으로 버전관리도 가능할 수준의 detail하고 substantial한 정보

# 링크

## Catalog.json

- 실질적인 bundle들의 정보로서 모든 addressables의 이름, 경로, Key, ResourceType이 정의 되어 있음
- 해당 파일을 이용하면 이전 catalog와의 diff를 통해 자체적으로 버전관리도 가능할 수준의 detail하고 substantial한 정보

# Managing catalogs at runtime

- 기본적으로 Addressables system은 런타임 중 자동적으로 카탈로그를 관리합니다.
- 런타임에 추가 카탈로그를 로드할 수 있습니다.

# Loading additional catalogs

- LoadContentCatalogAsync는 호스팅 서비스 또는 로컬 파일 시스템에서 추가 콘텐츠 카탈로그를 로드 합니다.
- 카탈로그 로드 작업이 완료되면 새 카탈로그의 키를 사용하여 주소 지정 로드 함수를 호출할 수 있습니다.
- 카탈로그 해시 파일을 카탈로그와 동일한 URL에 제공하는 경우 주소 지정은 보조 카탈로그를 캐시 합니다.
- 클라이언트 응용 프로그램은 나중에 카탈로그를 로드 할 때 해시가 변경된 경우에만 새 버전의 카탈로그를 다운 로드 합니다.
- 카탈로그를 로드하면 언로드할 수 없습니다.
- 그러나 로드된 카탈로그를 업데이트할 수 있습니다.

# Loading additional catalogs

- 카탈로그를 업데이트하기 전에 카탈로그를 로드한 작업의 작업 핸들을 해제해야 합니다.
- 일반적으로 카탈로그를 로드한 후 작업 핸들을 잡고 있을 이유가 없습니다.
- `autoReleaseHandle` 파라미터를 `true`로 설정하여 자동으로 해제할 수 있습니다.

# Updating catalogs

- 카탈로그 해시 파일을 사용할 수 있는 경우 Addressables는 카탈로그를 로드 할 때 해시를 확인하여 제공된 URL의 버전이 카탈로그의 캐시된 버전보다 최신인지 확인합니다.
- 원하는 경우 기본 카탈로그 검사를 끄고 카탈로그를 업데이트하려는 경우 Addressables.UpdateCatalogs 함수를 호출할 수 있습니다.
- LoadContentCatalogAsync를 사용하여 카탈로그를 수동으로 로드한 경우 카탈로그를 업데이트하기 전에 작업 핸들 해제해야 합니다.

# Updating catalogs

- UpdateCatalog 함수를 호출하면 작업이 완료될 때까지 다른 모든 주소 지정 가능 요청이 차단됩니다.
- 작업이 완료된 직후(또는 autoRelease 매개변수를 true로 설정) UpdateCatalogs에서 반환된 작업 핸들을 해제할 수 있습니다.
- 카탈로그 목록을 제공하지



# Reference

- <https://shkimo811.tistory.com/15> (P22 ~ P28)
- <https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html> (P29 ~ P)