**KeyedServices** is a layered component that is used on iOS as required for the app to be published on AppStore. Reason for that is because iOS does not allow the application to direct communicate with the hardware like Android, which is one of the reasons why iOS is always smoother on using in the long-term and Android devices tend to die faster. Each application can specifically request which layer of Services/Framework they want to use, and the OS will return exactly just that.

One of the main reasons KeyedServices exist is to manage the service/profile that is currently running. Most of the time, the service/profile will depend on other services that is running for it to run. What KeyedServices is doing is provide a notification of LiveObject or DestroyedObject so that other services can be communicate, understand the status and situation and know if it should run or not to avoid any unnecessary error.

***~/keyed\_service/content/browser\_context\_dependency\_manager.h***

**Type**: Singleton

**Purpose**: To get notification about a context destruction and forward the notification to each of the BrowserContextKeyedService so each of them know what to do.

* **void RegisterProfilePrefsForServices():** use to specify which profile they want to get notification about. Parameter registry will be insert into BrowserContext object as each profile is distinguished from each other’s. Because of this, the class has to be in Singleton in order to avoid any recreation object which replace the registry value
* **void CreateBrowserContextServices():** check if the registering profile has started service or not. Reason for using this is because there might be some profile depend on this so they can start without causing problem.
* **void CreateBrowserContextServicesForTest():** use this for profile testing purpose only
* **void DestroyBrowserContextServices():** use to destroy any profile or service that is depending on this. Some profiles or services are unable to restart or destroy itself, using this can alternatively destroy a non-working/non-using service.
* **Base::CallbackListSubscription RegisterCreateServicesCallbackForTesting():** derived the base function, using this function to show the list of depend profile/services related to this profile.
* **void AssertBrowserContextWasntDestroyed():** check if the context required to be destroy in DestroyBrowserContextServices has completely destroyed it. Depending on the value get from DCHECK\_IS\_ON() function, either NOTREACHED() (which mean the services has been destroyed) or DumpWithoutCrashing() (which mean the services hasn’t been destroyed) will be use.
* **void MarkBrowserContextLive():** mark the context as live, so that when the current profile trying to destroy a context, it knows about the state and avoid destroying a running context.
* **static BrowserContextDependencyManager\* GetInstance():** function to return the object state as in Singleton

***~/keyed\_service/content/browser\_context\_keyed\_service\_shutdown\_notifier\_factory.h***

**Type**: no type

**Purpose**: A Base class for KeyedServiceShutdownNotifier, probably used to notify other object connect to it when the current service is down.

***~/keyed\_service/content/browser\_context\_keyed\_service\_factory.h***

**Type: Observer**

**Purpose:** base class for BrowserContext that manage the services. It takes the BrowerContext and return service connecting to that to manage the status Live/Destroyed for each of the services. Each service derives from this must be a Singleton and also must specific which service it is depending from so they can be notified when it is ready to run.

* **void SetTestingFactory():** take testing\_factory object to create a KeyedService when context request. testing\_factory can be empty/null
* **KeyedService\* SetTestingFactoryAndUse():** similar to SetTestingFactory but testing\_factory object cannot be empty/null as it is required to return a KeyedService object to run.
* **KeyedService\* GetServiceForBrowserContext():** map the context to some necessary service that need to run with. If the services is required, create\_ will return true and trigger BuildServiceInstanceFor() if the services is not running. The context will send its object to the required service and have that service running on the designated context.
* **BrowserContextKeyedServiceFactory():** use to communicate with BrowserContextDependencyManager which control the context and the service running on it.
* **virtual content::BrowserContext\* GetBrowserContextToUse():** find the browser context to use
* **virtual bool ServiceIsCreatedWithBrowserContext():** some service is required to run as soon as the context is running so by using this, we can specify which service is required for which context and once the context is running, that service will also run to support it.
* **bool ServiceIsNULLWhileTesting():** set the target service to be NULL. Some service are required to run after the context is running but in the testing, the service can stay idle or quite until the testing is over.
* **virtual KeyedService\* BuildServiceInstanceFor():** return a KeyedService value for each of the service in the context
* **virtual void BrowserContextShutdown():** call every ServiceFactory and trigger the Shutdown(), where they will be removing dependencies with other service to be shutdown without error. At this stage, if everything works correctly, every service will be on their own, without depending on or relating to other services.
* **void BrowserContextDestroyed():** once Shutdown() has completed its task, the Destroyer will set the default implementation for all services and remove mapping + pointer from the context.

***~/keyed\_service/content/refcounted\_browser\_context\_keyed\_service\_factory.h***

**Type:**

**Purpose:** This is the same with browser\_context\_keyed\_service\_factory.h, however, while browser\_context\_keyed\_service\_factory.h ensure that all service running for the context is ThreadSafe(), the factory itself is not ThreadSafe() and might not be able to run. This context is running to ensure every context and service are ThreadSafe and be able to run smoothly

Video Link:

* <https://web.microsoftstream.com/video/49367e39-d39f-4d5c-b8e3-95829f6aead5> (13:44)
* <https://web.microsoftstream.com/video/babcb0ca-80ce-45c6-837d-0a4763ea4b45> (13:52)
* <https://web.microsoftstream.com/video/48744801-0c23-4bea-991c-1d33ae934598> (14:50)
* <https://web.microsoftstream.com/video/9ac70dc1-baa6-4bd1-ba2c-5d6a420c96e5> (17:01)
* <https://web.microsoftstream.com/video/8b312f12-e234-43ad-829e-9a1eb0b293e2> (17:49)
* <https://web.microsoftstream.com/video/0c68b6bd-dc50-462b-a615-eab460a63082> (15:54)
* <https://web.microsoftstream.com/video/8f6d2d45-e508-44c7-8523-f85e540cd6e5> (12:56)
* <https://web.microsoftstream.com/video/224547f6-9d10-44a0-a375-41cc5a2a5485> (16:41)
* <https://web.microsoftstream.com/video/8c1604ad-1fc7-44b8-9bca-92bad1c158b5> (22:34)
* <https://web.microsoftstream.com/video/b1c8c25e-dc0a-4e75-a163-e39bfb59daac> (18:37)