



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

# Web Services and Applications Deployment Guide

Caching

---

## Contents

- [1 Caching](#)
  - [1.1 GWS API Caching](#)
  - [1.2 GWS Platform API Caching](#)
  - [1.3 Frequently Asked Questions](#)

# Caching

In Genesys Web Services 8.6, caching is done in **GWS API Service** and **GWS Platform Service**, however, the caching mechanism is different for each service which is noted as follows:

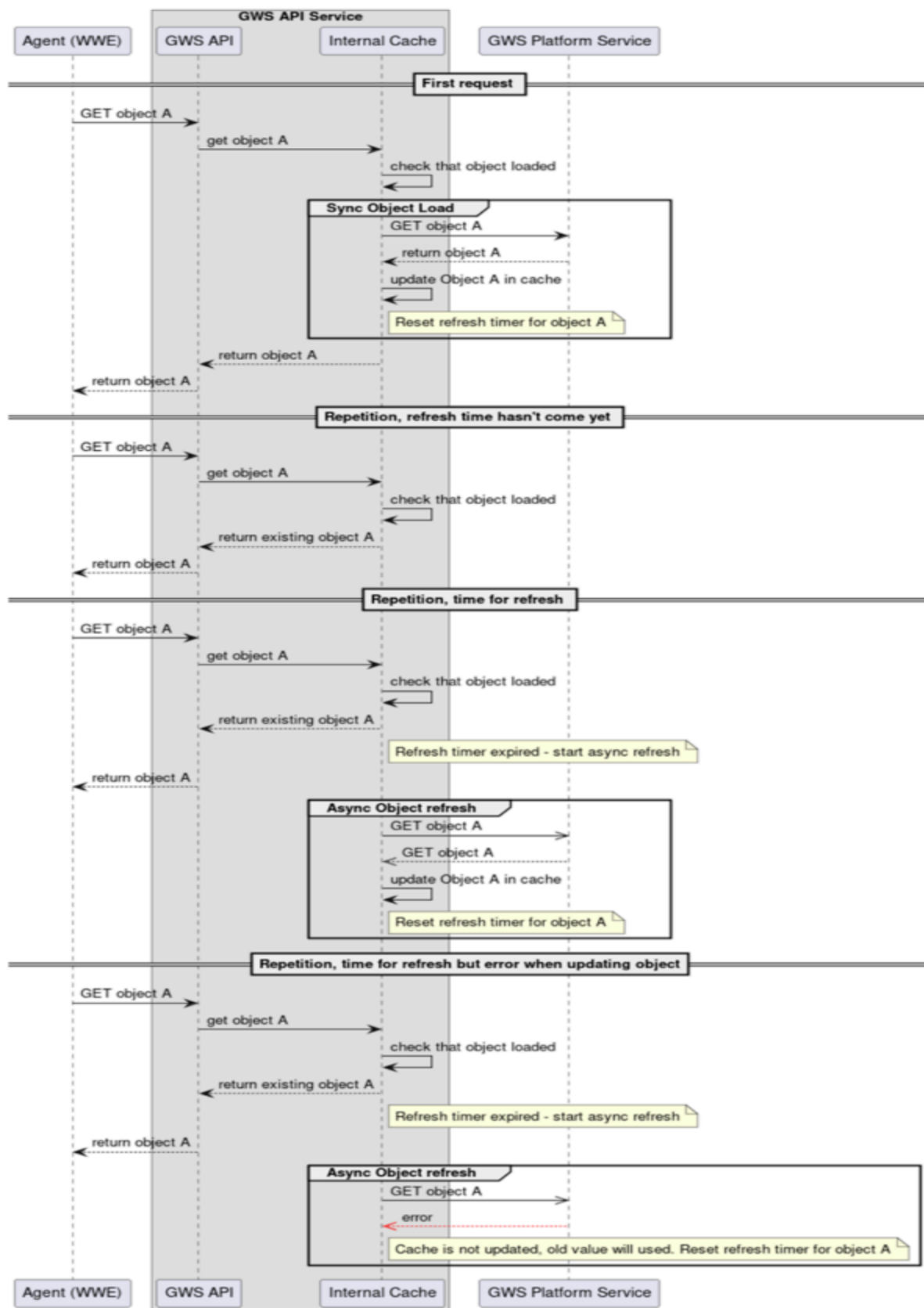
- **GWS API Service** implements its own small local cache.
- **GWS Platform Service** caches specific object types from Configuration database and keeps them in its own memory cache.

## GWS API Caching

GWS locally caches a small subset of configuration data. The maximum size of the cached subset of data is limited and by default it is set to 1000 objects of same type (users, DNS, places, and so on). Adding additional objects will depend on the Heap size. Each object will be refreshed after the refresh timeout has lapsed (default is 5 min) in the background. Meanwhile, the object stored in cache will be returned immediately without waiting for the refresh completion. Since the object refresh is triggered by read requests, the *stale* objects will not be refreshed until accessed, which can significantly reduce the load on Configuration Database and Platform API.

For each object type, the refresh rate can be configured using the corresponding option in the **cachingSettings** section of the **application.yaml** file, for example, **placeTtl**, **usersTtl**, and so on. For full list of caching options, see [Caching Settings](#).

### GWS API Caching Sequence diagram



### Important

As the contact center objects are cached at two levels, under normal circumstances, any object that is modified in Configuration Server will take a maximum of (API cache timeout + 1 second) to be reflected. If caching is disrupted for any unknown reason, then the Platform API's cache refresh will self-heal and restore the caching process.

### Important

On top of caching, GWS API utilizes Elasticsearch to fulfill API requests that require searching or filtering operations. API Configuration Indexing nodes add new objects or update existing during re-indexation procedure that happens on regular basis by schedule that is 15 minutes by default (indexVerificationInterval configuration parameter). Hence, if new object has been added to ConfigServer or something has been updated, those changes will be visible by GWS API only when next re-indexation procedure is completed. That behavior is different in compare with 8.5 where changes are applied immediately in primary region where API SyncNode is deployed. Non-primary regions in 8.5 deployments behave the same way as 8.6 now.

## GWS Platform API Caching

**GWS Platform Service** serves 'read API requests' using an internal in-memory cache or through Configuration Server. Note that not all object types are supported for caching. For the list of supported datatypes for which data is read directly from Configuration Server database, refer [here](#).

Unsupported types and write operations (create/update/delete) are still handled through Configuration Server. All reads from database are done in subsequent manner through a single database connection by background processes. The background processes that support these requests are:

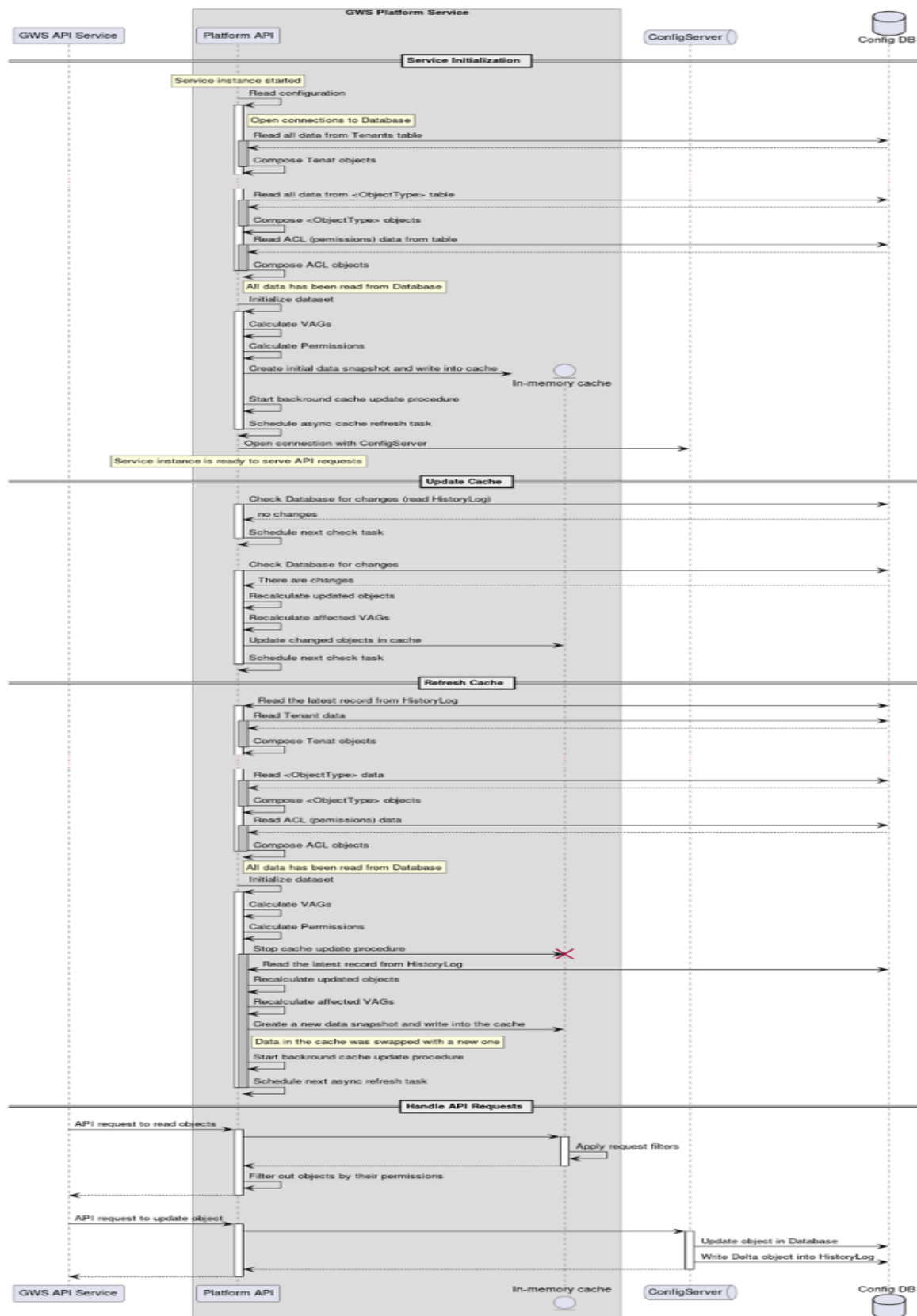
- A process for periodical full cache reevaluation that runs every 15 minutes (can be configured).
- A process for ongoing cache update procedure that runs every second picking up all changes in the database since the previous run, hence keeping the cache up-to-date. GWS Platform Service will retain and use existing cached information for the entire duration of any connectivity issues with the Configuration Database.

Supported object types for handling via in-memory cache are:

- CFGAccessGroup
- CFGActionCode
- CFGAgentGroup
- CFGAgentLogin

- CFGApplication
- CFGCallingList
- CFGCampaign
- CFGCampaignGroup
- CFGDN
- CFGDNGroup
- CFGEnumerator
- CFGEnumeratorValue
- CFGField
- CFGFilter
- CFGFolder
- CFGFormat
- CFGHost
- CFGPerson
- CFGPlace
- CFGPlaceGroup
- CFGRole
- CFGScript
- CFGService
- CFGSkill
- CFGSwitch
- CFGTableAccess
- CFGTenant
- CFGTransaction
- CFGTreatment

### **GWS Platform Caching Sequence diagram**



## Frequently Asked Questions

### GWS API Caching

#### **Do I have control over the cache refresh timer?**

Yes, you do have control over refresh timeouts, you can configure them using [Caching Settings](#).

#### **Will I know how old the data is in the response? If the refresh fails, will I be notified in the next request that the data is outdated?**

No, we don't provide any info on how old the data is since it might be composed from multiple data sources (caches) with different ages.

### GWS Platform API Caching

#### **What happens when a read API request is made and the configuration synchronization process fails more than once?**

GWS Platform Service isn't ready until cache is populated once. After that, all read requests for supported data types are handled only via in-memory cache which is updated by background processes. If the update procedure fails then a full refresh is scheduled immediately (with delay in action). Also, cache is refreshed on a regular basis, every 15 mins.