GVP Deployment Guide

How the Fetching Module Works

4/25/2025

How the Fetching Module Works

The Media and Call Control Platforms use the Fetching Module to fetch documents and perform caching. The Fetching Module maintains a high-performance in-memory cache and interfaces with the on board Squid Caching Proxy.

In GVP 8.1.2 and later releases:

- Fetching Module is no longer a separate component (integrated with the Media and Call Control Platforms).
- Squid is an optional component.

This section provides information about the following topics, to explain how the Fetching Module and the Squid Caching Proxy perform their role in a GVP deployment:

Caching

Unlike visual browsers, there are no end-user controls in the VoiceXML interpreter (the NGI) context to enable stale content to be updated or refreshed. Instead, the VoiceXML document itself enforces cache refreshes, through appropriate use of the maxage and maxstale attributes. However, these attributes interact with other proxy settings and HTTP cache-control mechanisms at various levels, as described in the following subsections.

Tip

The legacy VoiceXML interpreter, GVPi, does not use the Fetching Module to fetch documents or perform caching; instead it uses the Page Collector module. For a description of how the Page Collector fetches and performs caching, see Page Collector Caching.

Non-HTTP/1.1-Compliant Caching

The Fetching Module caches documents in-memory, in accordance with configurable maximum age and URL substring parameters (the iproxy.cache_max_age, iproxy.cache_error_max_age, and iproxy.no_cache_url_substr configuration parameters).

HTTP/1.1-Compliant Caching

The GVP 8.1.1 and earlier 8.x releases of the Fetching Module use a caching proxy (Third-Party Squid) for HTTP/1.1-compliant caching. Although the caching proxy generates HTTP/1.0 requests, it supports HTTP/1.1 caching functionality.

Starting in GVP 8.1.2, the Fetching Module is integrated with the Media Control and Call Control Platforms and is HTTP/1.1-compliant. In addition, the Squid Caching Proxy is optional.

The caching policies of the VoiceXML interpreter context adhere to the cache-correctness rules of HTTP/1.1. In particular, the Expires and Cache-Control headers are honored.

Caching Policies

- The application server maintainer/content provider can provide guidelines for content expiry by using the Cache-Control and Expires HTTP response headers.
- If these headers are not present, the Fetching Module (or, in GVP 8.1.1 and earlier 8.x releases, Squid) uses heuristics to generate expiry times.
- The application developer can control the caching behavior of application resources, by using the maxage and maxstale attributes for each URI-related VoiceXML tag. This behavior includes forcing a validation of the current cache contents (using maxage), and accepting expired cache contents (using maxstale).
- The platform maintainer can control cache-resource usage through the Media Control Platform or Call Control Platforms (or, in GVP 8.1.1 and earlier 8.x releases, Squid) configuration.

Caching Behavior

The primary effect of the caching policies is that the client has control over what it will accept from the cache, even if the server has specified an Expires header or maxage/maxstale attributes, or if the caching proxy has generated an expiry time itself.

- Documents from the web server will be delivered with none, one, or both of the response headers.
- If an Expires header is present, it is used to set the expiry time of the object in the cache.
- If the Expires header is not present, Squid applies a heuristic to set an expiry time.
- If a Cache-Control header is present in the response, it is used to control expiration times, and it overrides an Expiry time if it is specified.

Note: If the policy requires a fetch from the server, it is an optimization to perform a get if modified

(the request includes an If-Modified-Since [IMS] header) on a document that is still present in the cache. Squid performs this optimization.

maxage and maxstale

VoiceXML enables the application developer to control caching policy for each use of each resource.

The application developer can specify maxage and maxstale attributes for each resource-related element. These attributes provide fine-grained control over when documents are returned from the cache, and when they are fetched from the origin server. For example:

- Setting `maxage` to a nonzero value means that the Fetching Module might be forced to get a fresh copy of a resource that may not yet have expired in the cache. Setting maxage to zero (0) means that Squid is unconditionally forced to get a fresh copy.
- Using `maxstale` enables the application developer to specify that an expired copy of a resource that is not too stale (according to the rules of HTTP/1.1) can be used. This can improve performance by eliminating a fetch that would otherwise be required to get a fresh copy. This is especially useful for application developers who might not have direct server-side control of the expiration dates of large static files.

> **Tip**
>
> Like other caching proxies that support maxage and maxstale, the Fetching Module does not delete items from the cache after their expiry time, unless other cache requirements (such as memory or disk-usage limits) dictate such action. The reason for this is that the client might specify that an expired resource is acceptable.
>
> Some resources may be addressed by URIs that name protocols other than HTTP, and that do not support the maxage and maxstale attributes. If the protocol does not support the concept of resource age, the interpreter context computes the age of a resource from the time that it was received. If the protocol does not support the concept of resource staleness, the interpreter context treats the resource as having expired immediately upon receipt.

The `maxage` and `maxstale` attributes interact with server-provided expiry times to produce a variety of caching behaviors. The table below describes some sample behaviors.

Table: Using maxage and maxstale Attributes

Desired behavior	maxage	maxstale	Notes
Client control over expiry	**<desired_expiry>**	`0`	• Caching is based on the Expires header. • Refetch is based on maxage and maxstale.

Desired behavior	maxage	maxstale	Notes
			• Uses IMS.
Expired document acceptable	**<large_value>**	**<desired_maxstale>**	• Caching is based on the Expires header. • Refetch occurs after Expiry time plus maxstale. • Uses IMS.

maxage, maxstale, and the Initial Page

For the initial page request, the GVP session ID is submitted as part of the URL. Because this ID is unique, the requested URL appears unique; therefore, the maxage and maxstale parameters have no meaning for that page. However, they do have meaning for the initial root page.

Configuration parameters `vxmli.initial_request_maxage` and `vxmli.initial_request_maxstale` in the Media Control Platform set the values of the `maxage` and `maxstale` parameters for the initial root page. For both parameters, the default value is -1 (undefined).

Determining Expiry Time

Web servers may or may not return an Expires response header to the client.

- If the Web server does return an Expires response header, this expiry time is used in the cache refresh algorithm.
- If, instead, the Web Server provides expiration information as part of a Cache-Control header (using maxage/maxstale), this information will be used to control cache expiry.

Expiration Model

The Fetching Module uses a refresh-rate model, instead of a time-based expiration model. Objects are not purged from the cache when they expire. Instead of assigning a time to live when the object enters the cache, the Fetching Module checks freshness requirements when objects are requested:

- If an object is fresh, it goes directly to the client.
- If an object is stale, an If-Modified-Since request is made and sent to the web server.

For information about how to use HTTP caching to improve performance, see Optimizing Performance through HTTP Caching.

Squid Configuration File

The Squid configuration file (`C:\squid\etc\squid.conf` for Windows and `<Directory>/etc/squid/`

`squid.conf` for Linux) controls configuration of the caching proxy. The configuration file is a text file that contains pairs of keywords and values (with no equal sign [=] between them). For example, the following pair defines port 3128 as the TCP port that the caching proxy will use for receiving requests:

```
http_port 3128
```

In general, the default Squid configuration file should be suitable for most installations. However, you might need to modify it for the following reasons:

- You need to configure for a second-level proxy.
- You cannot configure your Web Server to deliver Expires headers, and you want to change the Squid defaults for the expressions that Squid tries to match in SIP request-URI headers to control refresh behavior.
- You need to configure nonstandard safe ports or SSL ports for HTTP and SSL.

For more information about modifying the Squid configuration file, see the section about configuring the Squid caching proxy in the Genesys Voice Platform 8.5 User's Guide.

For detailed information about all Squid configuration items, see the Squid Configuration Guide at the ViSolve Squid Support Service website.

Changes to the Squid configuration file do not take immediate effect in the running configuration.

Squid Log Files

The caching proxy logs can provide useful information to help you identify performance issues or resolve VoiceXML or CCXML application problems.

Access Logs

The Squid access.log file is in the following location:

`C:\squid\var\logs\ (Windows) or /var/log/squid (Linux).`

The access log contains one entry for each HTTP (client) request and each Inter-Cache Protocol (ICP) Query. HTTP requests are logged when the client socket is closed. The native access.log file has ten fields. A single hyphen (-) indicates unavailable data.

For detailed information about the fields in the Squid access.log file, see the caching reference information appendix in the Genesys Voice Platform 8.5 User's Guide.

For information about how to schedule log rotations and manage the cache manually, see Managing the Cache.