# Guide Détaillé des Options gwd (GeneWeb Daemon)

Date de création : 11 septembre 2025
Outil : gwd - serveur HTTP/CGI de GeneWeb
Version analysée : branche master
Documentation technique exhaustive

---

## Table des matières



---

## 1. Vue d'ensemble

**gwd** est le serveur web de GeneWeb. Il répond aux requêtes HTTP, rend les pages dynamiques de la base, sert des assets (images/CSS/JS), gère l'authentification « wizard/friend », des jetons d'accès temporaires, la protection anti-robots, et charge des plugins.

- Modes d'exécution:
  - Serveur HTTP natif (par défaut)
  - Mode CGI forcé (`-cgi`)
- Entrée: requêtes HTTP GET/POST
- Sortie: HTML/JSON/Images/Assets

---

## 2. Architecture et mode de fonctionnement

```
(* Point d'entrée principal *)
main ()
   -- Parse des options (Arg)
```

```
   -- Chargement/ordre des plugins (-plugin/-plugins)
   -- Initialisation GWPARAM & cache_lexicon
   -- Mode CGI ? => geneweb_cgi | sinon => geneweb_server

(* Serveur HTTP *)
geneweb_server -> Wserver.start ~addr ~port ~n_workers (connection)

(* Traitement d'une requête *)
connection (addr, request) script_name contents
   -- Filtre robots.txt, gwd.xcl, -only
   -- build_env (multipart/form-data | urlencoded)
   -- conf_and_connection (make_conf + Request.treat_request)
```

- Le serveur peut être multi-processus (`-daemon`) et multi-workers (`-n_workers`).
- En CGI, les en-têtes sont reconstruits depuis les variables d'environnement (`QUERY_STRING`, `REQUEST_METHOD`, etc.).

---

## 3. Démarrage, sockets, CGI et environnement

- **Binding**: `-a <ADDR>` et `-p <PORT>` définissent l'adresse/port d'écoute.
- **Sockets (Windows)**: `-wd <DIR>` configure les fichiers `gwd.sin/gwd.sou` et `cnt_dir`.
- **CGI forcé**: `-cgi` (optionnelle `-cgi_secret_salt <SALT>` pour signer les formulaires; cf. Auth).
- **Répertoires**:
  - `-hd <DIR>`: préfixe assets (`etc`, `images`, `lang`) et ajout au chemin sécurisé.
  - `-etc_prefix`, `-images_prefix`, `-images_dir`: surcharges fines des chemins.
  - `-bd <DIR>`: emplacement des bases (par défaut `bases`).
- **Redirection**: `-redirect <ADDR>` répond avec une page indiquant la nouvelle adresse du service.
- **Fichier d'exclusion**: `gwd.xcl` (mêmes base-name que binaire), motifs avec `*` pour refuser des hôtes.

---

## 4. Authentification, sessions et sécurité

### 4.1 Rôles et schémas

- **Wizard** (admin) et **Friend** (ami) via mot de passe (ou fichiers d'auth).
- Schémas pris en charge:
  - **Basic** (par défaut)
  - **Digest** (`-digest`)

  - **Token** (jetons temporaires stockés dans `actlog`, utilisés hors CGI pour passer l'accès dans l'URL)

### 4.2 Paramètres et fichiers

- **Ligne de commande**: `-wizard <PWD>`, `-friend <PWD>`
- **Fichiers** (recommandés en CGI): via `.gwf/.cgf` par base: `wizard_passwd_file`, `friend_passwd_file`
- **Fichier global**: `-auth <FILE>` (limité, non compatible CGI) avec lignes `user:password`
- **Timeout**: `-login_tmout <SEC>` (durée des entrées avec mots de passe en CGI)
- **Trace échecs**: `-trace_failed_passwd` ajoute l'empreinte d'échec dans les logs

### 4.3 Jetons et accès

- Paramètre de requête `w=`:
  - `w=w` ou `w=f` => initie l'auth (Basic/Digest)
  - `w=<token>` => accès par jeton (hors CGI)
- Génération: `set_token` enregistre `(from_addr, base_pw)` dans `actlog` (protégé par lock `gwd.lck`).
- Expiration: basée sur `-login_tmout`; purge à la volée.

### 4.4 Mode digest et secret_salt

- En mode Digest (`-digest`), un `nonce` est calculé; rejet si non-cohérent.
- En CGI, `-cgi_secret_salt <SALT>` peut être utilisé pour signer des formulaires.
- En mode serveur, un `SECRET_SALT` est partagé aux workers (via env) pour synchroniser les digests.

### 4.5 Visibilité et RGPD (paramètres de base)

- Paramètres par base (fichier `.gwf`), ex: `semi_public`, `private_years`, `hide_private_names`… influent sur les vues visiteurs.

---

## 5. Plugins: chargement, intégrité, assets

- **Chargement ciblé**: `-plugin path/to/foo` => charge `plugin_foo.cmxs`
- **Chargement en lot**: `-plugins path/to/plugins/` => charge tous les plugins en respectant l'ordre via `META` (dépendances résolues; topologique)
- **Intégrité**: par défaut, vérification via `GwdPluginMD5.allowed` (whitelist officielle)
  - `-unsafe` (paramètre additionnel à passer juste après `-plugin`/`-plugins`) désactive la vérification

  - `-force` active le plugin pour toute base
  - Exemple: `-plugins ./plugins -unsafe -force`
- **Assets plugin**: dossier `assets/` (ex: `assets/lex/` enrichit les lexiques); servi via routes statiques (CSS/JS/fonts/images)

  Avertissement: `-unsafe` permet d'exécuter du code non vérifié. À utiliser uniquement en développement ou avec des sources sûres.

---

## 6. Fichiers statiques, images et lexiques

### 6.1 Images

- Paramètres: `-images_prefix`, `-images_dir`
- Requêtes `m=IM&v=<fname>` -> `ImageDisplay.print_image_file`
- Chemins résolus par `Image.path_of_filename` et le préfixe configuré

### 6.2 Fichiers statiques (CSS/JS/Fonts/Maps)

- Recherche dans l'ordre:
  1) chemins des plugins autorisés (`assets/...`)
  2) répertoire `etc` de la base (`<base>/etc/...`)
  3) assets embarqués (distribution)
- Types servis: `.css`, `.js`, `.map`, `.otf`, `.ttf`, `.eot`, `.woff`, `.woff2`, `.svg`, `.png`, `.cache.gz`
- En-têtes: `Content-type`, `Content-length`, `Content-disposition`, `Cache-control` (1 an)

### 6.3 Lexiques (localisation)

- **Chargement**: `lang/lexicon.txt` + alias `lang/alias_lg.txt`
- **Cache**: `-cache_langs fr,en` pré-charge en mémoire
- **Ajout**: `-add_lexicon <FILE>` (multi-fichiers possible)
- Le lexique est influencé par les plugins actifs (ex: leurs `assets/lex`).

---

## 7. Protection anti-robots et journalisation

### 7.1 Robots

- `robots.txt` généré ou servi depuis l'asset `robots`
- **Exclusion dynamique**:
  - `-robot_xcl <CNT>,<SEC>` déclenche un blocage si trop de requêtes sur fenêtre glissante
  - Seuil minimal logué: `-min_disp_req <INT>`
- **Liste d'exclusion**: `gwd.xcl` (motifs `*`) pour refuser certains hôtes

### 7.2 Logs

- Destination: `-log <FILE>` (`-`/`<stdout>`/`<stderr>` supportés)
- Verbosité: `-log_level <N>` (0–7), `-debug` active backtrace, warnings runtime et niveau 7
- Traces d'accès: referer, user-agent, utilisateur (wizard/friend), lenteur requêtes (voir env `GWD_SLOW_QUERY_THRESHOLD`)

---

## 8. Options détaillées (par catégorie)

### 8.1 Chemins et ressources

- `-hd <DIR>`: préfixe pour `etc`, `images`, `lang` et ajout à la zone sécurisée d'assets
- `-etc_prefix <DIR>`: surcharge du chemin `etc`
- `-images_prefix <DIR>`: surcharge du préfixe `images` (ex: `file://...` construit automatiquement si `-images_dir`)
- `-images_dir <DIR>`: chemin relatif pour repérer physiquement les images; converti en `file://` absolu
- `-bd <DIR>`: répertoire des bases (défaut: `bases`)

### 8.2 Réseau et serveur

- `-a <ADDR>`: adresse d'écoute (défaut: toutes)
- `-p <PORT>`: port d'écoute (défaut: 2317)
- `-daemon` (Unix): mode démon (fork initial + redirection stdout/err -> /dev/null)
- `-n_workers <NUM>` (Unix): nombre de workers (défaut: 20)
- `-max_pending_requests <NUM>` (Unix): file d'attente maximale (défaut: 150)
- `-conn_tmout <SEC>` (Unix): timeout connexions (0 = illimité)
- `-max_clients` (Unix): obsolète; utiliser `-n_workers` et `-max_pending_requests`
- `-no-fork` (Unix): obsolète; équivalent à `-n_workers 0`
- `-wd <DIR>`: dossier sockets (Windows) et compteur d'accès
- `-no_host_address`: pas de reverse DNS, logue l'IP brute
- `-only <ADDR>`: seule adresse autorisée (multiples utilisations possibles)
- `-redirect <ADDR>`: répond que le service est déplacé vers `<ADDR>`

### 8.3 Langue et affichage

- `-lang <LANG>`: langue par défaut (sinon détectée via `LANG/LC_CTYPE` ou `-blang`)
- `-blang`: privilégier la langue du navigateur si disponible (`Accept-Language`)
- `-cache_langs fr,en,...`: pré-charger les lexiques
- `-add_lexicon <FILE>`: ajouter des entrées au lexique
- `-setup_link`: afficher un lien vers gwsetup en bas des pages

- `-allowed_tags <FILE>`: lister les balises HTML autorisées (une par ligne)

### 8.4 Authentification et accès

- `-wizard <PASSWD>`: mot de passe Wizard
- `-friend <PASSWD>`: mot de passe Friend
- `-wjf`: Wizard considéré comme Friend (mode « just friend » permanent)
- `-digest`: schéma Digest HTTP (plus sûr que Basic)
- `-auth <FILE>`: fichier global `user:password` (non recommandé en CGI)
- `-login_tmout <SEC>`: expiration des jetons/entrées CGI
- `-trace_failed_passwd`: journaliser les échecs d'auth (sauf Digest)
- `-nolock`: ne pas verrouiller les fichiers lors des écritures (déconseillé)

### 8.5 Plugins

- `-plugin <DIR>`: charger `plugin_<DIRNAME>.cmxs`
  - modificateurs immédiats: `-unsafe` et/ou `-force`
- `-plugins <DIR>`: charger tous les plugins du répertoire (résolution `META` + dépendances)
  - modificateurs immédiats: `-unsafe` et/ou `-force`

### 8.6 Mode CGI et secrets

- `-cgi`: forcer le mode CGI (sinon serveur HTTP)
- `-cgi_secret_salt <STRING>`: sel secret pour signatures/nonce (CGI)

### 8.7 Journalisation et version

- `-log <FILE>`: destination des logs (`-`, `<stdout>`, `<stderr>`)
- `-log_level <N>`: niveau de verbosité (défaut courant affiché à l'aide)
- `-debug`: mode debug (verbeux maximal + backtraces)
- `-version`: afficher version, source, branche, dernier commit et quitter

### 8.8 Cache et performances (Unix)

- `-cache-in-memory <DATABASE>`: précharger une base en mémoire (si support `Gw_ancient`)
- `-predictable_mode`: désactive certains aléas (hasard/cache) pour des tests reproductibles (ATTENTION: non recommandé en production)

---

## 9. Variables d'environnement et fichiers auxiliaires

### 9.1 Variables d'environnement

- `GWD_SLOW_QUERY_THRESHOLD`: seuil (secondes) d'une requête considérée lente (log niveau « warning »)

- `SECRET_SALT`: sel secret partagé aux workers; fixé par le master, lu par les workers (ne pas supprimer)
- En CGI: `QUERY_STRING`, `REQUEST_METHOD`, `CONTENT_TYPE`, `CONTENT_LENGTH`, `HTTP_*` (reconstruit dans `geneweb_cgi`)

### 9.2 Fichiers auxiliaires

- `actlog`: stockage des jetons d'accès (protégé par `gwd.lck`)
- `gwd.lck`: verrou pour opérations atomiques (ex: actlog, robot check)
- `STOP_SERVER`: fichier drapeau pour ordonner l'arrêt (dans `cnt_dir`)
- `gwd.xcl`: motifs d'exclusion d'hôtes
- `robots`: modèle de `robots.txt` si présent dans assets

---

## 10. Performances, cache et mode prédictible

- **Workers/queue**: ajuster `-n_workers` et `-max_pending_requests` selon charge/concurrence
- **Timeouts**: `-conn_tmout` (Unix) et `-login_tmout` (CGI) selon besoins
- **Cache lexiques**: `-cache_langs` pour éviter coûts I/O
- **Préchargement DB**: `-cache-in-memory` (si disponible) pour bases très consultées
- **Mode prédictible**: `-predictable_mode` désactive l'aléa (tests), pénalise la sécurité (à proscrire en prod)
- **Seuil lenteur**: `GWD_SLOW_QUERY_THRESHOLD` logue les requêtes lentes (profilage)

---

## 11. Exemples d'utilisation

### 11.1 Lancement simple (local)

```
gwd -bd bases -p 2317 -hd gw -log <stdout> -log_level 6
# => http://localhost:2317/ma_base
```

### 11.2 Mode CGI

```
# Dans un hôte CGI/FCGI, forcer le mode CGI
gwd -cgi -bd bases -hd gw -cgi_secret_salt "my-secret"
```

### 11.3 Authentification et jetons

```
# Wizard + Friend + schéma Digest
gwd -wizard secretW -friend secretF -digest

# Lien d'accès par jeton (hors CGI), côté client
```

```
# 1) requête initiale avec w=w ou w=f (déclenche l'auth)
# 2) gwd renvoie (ou positionne) un token, réutilisable pendant -login_tmout
```

### 11.4 Plugins officiels et répertoire d'assets

```
# Charger plugins depuis ./plugins, vérifier intégrité, forcer activation pour toutes bases
gwd -plugins ./plugins -force

# Développement local (non recommandé en prod)
gwd -plugin ./my_plugin -unsafe
```

### 11.5 Filtrage d'accès / anti-robots

```
# Accepter uniquement une adresse hôte, bloquer floods > 100 req / 10 s
gwd -only myhost.local -robot_xcl 100,10
```

### 11.6 Chemins images personnalisés

```
# Servir des images depuis un répertoire spécifique (résolu en file://...)
gwd -images_dir ./hd/images
```

---

## 12. Dépannage

### Port déjà utilisé

`Error: the port 2317 is already used …`

- Solution: arrêter l'autre service ou choisir un autre port avec `-p`.

### Droits insuffisants sur port Unix (< 1024)

`Error: invalid access to the port 80 …`

- Solution: exécuter en root ou utiliser un port > 1024.

### Incompatibilités CGI

- `-auth` et `-digest` affichent un avertissement en mode `-cgi`. Préférer les mots de passe/fichiers dans la configuration de base (`.gwf/.cgf`).

### Problèmes de plugins

- Échec de chargement Dynlink (`Register_plugin_failure`): vérifier `META`, dépendances, et intégrité (`GwdPluginMD5.allowed`), ou relancer avec `-unsafe` (développement).

**Accès refusé et logs**

- Utiliser `-trace_failed_passwd` (sauf Digest) et consulter `trace_auth.txt` (si activée), `gwd.lck`, et `actlog`.

---

## 13. Bonnes pratiques

- Production:
  - Éviter `-unsafe`, `-nolock`, `-predictable_mode`
  - Définir `-log` et un `-log_level` adapté (ex: 5–6), surveiller les requêtes lentes via `GWD_SLOW_QUERY_THRESHOLD`
  - Ajuster `-n_workers`, `-max_pending_requests`, `-conn_tmout`
  - Centraliser les mots de passe dans les fichiers par base, pas en CLI
- Sécurité:
  - Préférer `-digest` à `Basic`; en CGI, utiliser `-cgi_secret_salt`
  - Restreindre l'accès avec `-only`, `gwd.xcl`, et `-robot_xcl`
- Maintenance:
  - Utiliser `-plugins` avec META et whitelist (sans `-unsafe`) pour tracer les dépendances
  - Cacher les lexiques courants `-cache_langs` et, si supporté, précharger la base

---

Document basé sur l'analyse complète de `bin/gwd/gwd.ml` et `bin/gwd/README.md`. Cette documentation couvre l'intégralité des options exposées et leur comportement interne, avec recommandations de sécurité et d'exploitation.