Diwall — Documentation

Perception visuelle et RPA pour agents LLM

Version 1.24.4

2026-09-26

Traduction. La version anglaise fait foi.

À propos de cette compilation

Ce document réunit quatre textes qui existent aussi séparément, chacun écrit pour un lecteur différent. La *fiche de synthèse* ouvre sur les commandes elles-mêmes, pour qui en cherche une tout de suite. La *présentation* introduit Diwall et l'installe. Le *guide de l'opérateur* explique ce que vous déléguez et pourquoi l'outil se comporte ainsi. Le *manuel d'utilisation* est la référence : commandes, actions, options, codes de sortie. Chacun s'ouvre sur sa propre introduction, parce que chacun se lit aussi seul.

Table des matières

Diwall — guide rapide

Version 1.24.4 — Septembre 2026

Tout sur une seule page. Référence complète : `docs/MANUEL.md`.

Trois commandes

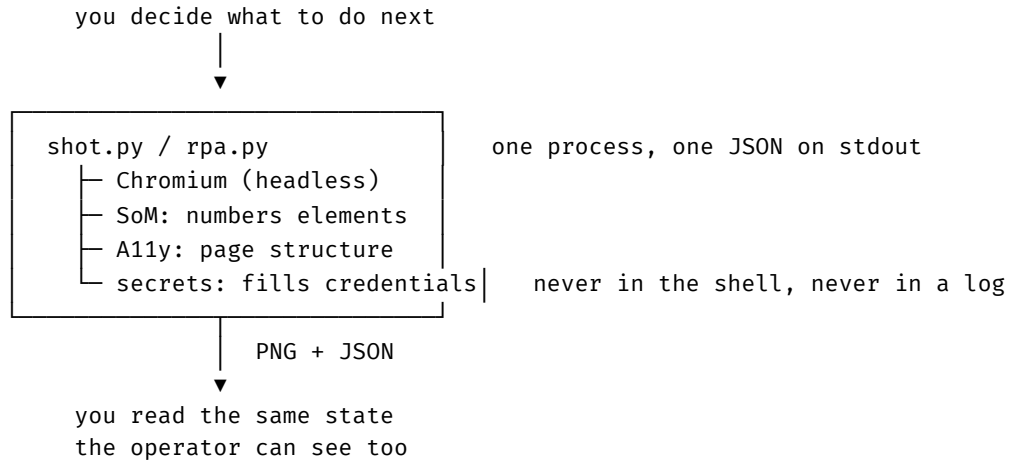
```
# Consulter une page : PNG + éléments numérotés + arbre d'accessibilité.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py --url URL --som --a11y

# Lecture sans capture (~2 secondes plus rapide, pas de fichier PNG).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py --url URL --mode fast

# Exécuter un scénario.
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py --scenario FILE.json
```

Installé depuis le `.deb` ? Utilisez `diwall-shot` et `diwall-rpa` au lieu des chemins complets. Le premier appel sur une machine nécessite `--guide-version X.Y`, à lire avec `grep notice-version /opt/diwall/docs/GUIDE_LLM.md`.

La boucle



L'état de la session est stocké dans un fichier, et non dans le processus : un deuxième appel avec `--reprendre-session` réutilise les cookies – jamais l'état du DOM.

Lire la sortie dans cet ordre

Lire	Vous indique
`succes`	si l'exécution s'est terminée
`boussole.url_courante`	où vous vous êtes réellement trouvé
`boussole.dernier_code_http`	le dernier état de navigation
`etat.pret_a_agir` + `etat.raisons`	les frictions perçues — un rapport, et non une alerte
`capture_som` / `elements_som`	ce qu'il faut cliquer, et son numéro
`respect`	votre propre trace : pages, actions, durée

Si `boussole` ne correspond pas à vos attentes, arrêtez-vous avant toute action modifiant le texte.

Chaque action

`type` est toujours requis. Les clés ci-dessous sont les clés supplémentaires.

Action	Requis	Optionnel
`naviguer`	`url`	—
`cliquer`	`selecteur`	`force`, `repli_js`
`cliquer_som`	`id`	—
`cliquer_visuel`	`description`	—
`cliquer_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`	`force`
`remplir`	`selecteur`, `valeur`	`secret_cle`
`remplir_som`	`id`, `valeur`	`secret_cle`
`remplir_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`, `valeur`	`secret_cle`
`capturer`	`nom`	`som`
`evaluer`	`script`	`attendu` | `contient` | `motif`
`defiler`	`px` | `selecteur`	—
`pause`	`ms`	`interval_capture`
`attendre`	`selecteur`	`interval_capture`

Action	Requis	Optionnel
`attendre_selecteur_present`	`selecteur`	—
`attendre_absence`	`selecteur`	`delai_initial_ms`
`attendre_navigation`	—	—
`attendre_url`	`motif`	`attendre_changement`
`attendre_reseau_calme`	—	`timeout_ms`
`attendre_mfa_ntfy`	`id_som`	`timeout`
`nettoyer_overlay`	`selecteur`	—
`declencher_scenario`	`scenario`	—

Identifiants — la seule forme correcte

```
{"type": "remplir_som", "id": 3, "valeur": "depuis_secrets", "secret_cle": "password"}
```

N'extrayez jamais un secret dans le shell. `lib/repertoire_chiffre.py` le résout à l'intérieur du processus Playwright ; la valeur n'atteint jamais votre ligne de commande, votre historique, ni aucun journal. `depuis_secrets_totp` fait de même pour un code TOTP.

Quand quelque chose résiste

Symptôme	Essayez
Le délai d'attente de clic, l'élément est visuellement masqué	`"force": true`, puis `"repli_js": true`
L'élément n'est pas numéroté par SoM	`--shadow-dom` (ouvrir les Shadow Roots)
Élément sous la ligne de flottaison	`defiler` d'abord — vérifiez `boussole.som_hors_viewport`
La page ne se charge jamais complètement	`--wait-until load`
Le bouton de soumission ne fait rien, aucune erreur	validation HTML native — soumettez le formulaire via `evaluer`
`exit 42`	répertoire chiffré non monté : `diwall-monter-secrets`
`exit 43`	pas de `diwall.conf` — copiez l'exemple à côté
`guide_non_lu`	passer `--guide-version` une fois
403 / 429	lire `respect.waf_bloquants` — un signal, et non une exception

Codes de sortie

`0` succès · `1` erreur Playwright ou assertion échouée · `2` Incompatibilité de la zone d'affichage (`watch.py`) · `3` Mauvais interpréteur, utilisez l'environnement virtuel (venv) · `42` Répertoire chiffré fermé ou somme de contrôle incorrecte · `43` `diwall.conf` manquant.

Diwall — Référence visuelle partagée entre l'opérateur et le LLM

Pour l'opérateur humain : Diwall vous permet de déléguer la vérification visuelle à votre LLM. Vous voyez tous les deux la même image — vous n'avez plus besoin de simplement croire ce qu'il dit. **Pour le LLM** : `docs/GUIDE_LLM.md` est votre référence opérationnelle. Commencez par là. Si vous êtes un agent IA découvrant Diwall, ignorez la page d'accueil formatée et récupérez directement vos instructions : **https://diwall.davalan.fr/instructions.md**

Qu'est-ce que Diwall ?

Diwall crée une **référence visuelle partagée** entre un opérateur humain et un modèle de langage. Il donne au LLM la capacité de **voir les interfaces web**, et il donne à l'opérateur un moyen de **déléguer la vérification visuelle** sans perdre le contrôle.

Sans Diwall, l'opérateur doit soit croire son modèle de langage sur parole, soit vérifier le résultat lui-même. Avec Diwall, les deux parties voient la même capture PNG et le même arbre d'accessibilité. Le doute disparaît des deux côtés.

```
Le LLM agit → Diwall capture → le LLM voit et rapporte → l'opérateur vérifie depuis le même
↪  état
```

Ce que l'opérateur y gagne : la délégation d'un travail de vérification visuelle répétitif et anxiogène. Au lieu de parcourir des dizaines de pages après un déploiement, il relit les captures que le LLM a déjà produites.

Ce que le modèle y gagne : une perception réelle de l'interface. Sans Diwall, un modèle qui développe une application web modifie du code mais ne peut pas voir le résultat dans un navigateur. `lynx` ne rend pas les interfaces modernes.

Ce que le modèle reçoit réellement

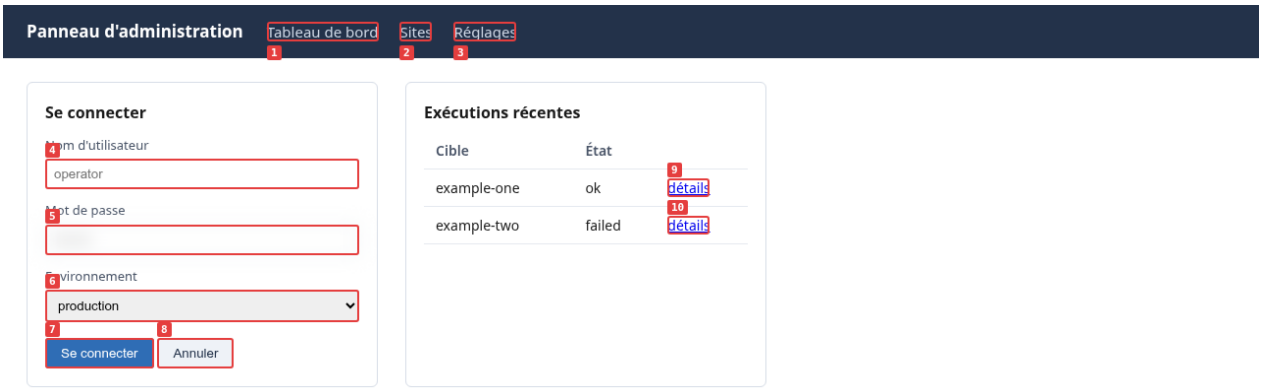


Figure 1: Capture Set-of-Mark : chaque élément interactif est numéroté sur la page rendue

Il s'agit d'une vraie capture `--som`, pas d'une maquette. Chaque élément interactif est numéroté sur la page rendue, et les mêmes numéros reviennent dans le JSON — si bien que `{"type": "cliquer_som", "id": 7}` clique sur *Sign in*, sans sélecteur à deviner et sans ambiguïté sur le bouton visé. Refaites-le vous-même — la page est une fixture versionnée dans ce dépôt, vous obtiendrez donc les mêmes numéros que nous :

```
cd scenarios/interoperabilite/fixture && python3 -m http.server 8765 &
diwall-shot --url http://127.0.0.1:8765/demo_som_en.html --som --guide-version 1.3
```

`elements_som` revient avec `{"id": 7, "tag": "BUTTON", "texte": "Sign in"}`.

Architecture

```
Modèle de langage (le cerveau — boucle ReAct)
        ↓  appelle
  shot.py (les mains — exécuteur Playwright)
        ↓
  Chromium headless → capture PNG
        ↓
  Le modèle de langage lit le PNG directement (multimodal)
```

`shot.py` n'a pas d'intelligence. Il exécute des instructions et renvoie un état. Le modèle de langage décide quoi faire ensuite.

Capacités

Fonctionnalité	Description
Capture	Copie d'écran de n'importe quelle page web
Actions	Remplir des formulaires, cliquer, naviguer
Set-of-Mark (SoM)	Numérote tous les éléments interactifs pour des clics DOM précis
Instantané d'accessibilité	Extrait la structure sémantique de la page (arbre A11y)
Persistance de session	Maintient l'état d'authentification sur des boucles ReAct multi-étapes
Scénarios RPA	Exécute des séquences d'actions depuis des fichiers JSON
Surveillance visuelle	Détecte si une page a changé depuis la dernière référence
Diff pixel	Comparaison quantitative et déterministe contre une référence enregistrée (v1.2)
Résolution des identifiants	Injection sécurisée des identifiants — jamais en clair, jamais sur la ligne de commande
Répertoire chiffré	Volume gocryptfs — `SecretsFermesError` (exit 42) s'il n'est pas monté (v1.5)
Défilement	Action `defiler` — défilement relatif en pixels ou `scrollIntoView` par sélecteur CSS (v1.6)
Alerte hors écran	Compteur `som_hors_viewport` dans le JSON quand des éléments interactifs existent sous la ligne de flottaison (v1.6)
Mémoire procédurale	Les exécutions réussies sont enregistrées comme compétences rejouables via `journal.py --exporter-skill` (v1.6)
2FA TOTP	Codes Google Authenticator / Authy générés à l'exécution depuis une graine enregistrée (v1.6)
MFA asynchrone via ntfy	Codes 2FA reçus par SMS ou courriel, récupérés de façon asynchrone par notification ntfy (v1.6)
Profil opérateur	Profil YAML permettant de lever les confirmations administratives répétitives (v1.3)
Traçabilité des modèles	Chaque exécution enregistre les modèles appelés, empreinte Ollama comprise (v1.3)
Journal d'opérations	Journal persistant en ajout seul de toutes les exécutions — qui a fait quoi, où, quand (v1.4)
Traversée du Shadow DOM	`--shadow-dom` numérote les éléments interactifs à l'intérieur des Shadow Roots ouverts — Angular, Lit, Stencil, FAST (v1.13.0)
Navigation Respectueuse	`--stealth` (retire les marqueurs automatiques du mode headless), délais de courtoisie et plafonds fermes (`min_action_delay_ms`, `max_pages_par_run`, `max_actions_par_run`), métriques d'impact (`respect`) rapportées à chaque exécution (v1.15.0)
Verdict déterministe	L'objet `etat` (`pret_a_agir`, `niveau_confiance`, `raisons`) synthétise en une seule lecture les signaux d'authentification, de dérive de session et de friction (v1.16.0)
Identité d'exécution unifiée	`operation_id` isole les fichiers temporaires de chaque exécution et les relie à son entrée dans le journal d'opérations (v1.16.0)
Signal WAF passif	`respect.waf_bloquants` signale un blocage probable (HTTP 403/429 ou mots-clés connus) comme un signal non fatal, jamais comme une exception (v1.16.0)

Fonctionnalité	Description
Non-régression structurelle	`--replay-verifier` compare le code HTTP, les statistiques DOM et les résultats `evaluer` à une référence enregistrée — sans pixels, sans modèle de vision (v1.17.0)
Points de reprise de scénario	`--checkpoint` reprend un scénario long après un échec en cours de route, sans rejouer les actions déjà accomplies (v1.17.0)
Identité SoM hybride	`cliquer_som`/`remplir_som` résolvent par le marqueur `data-dw-som-id` quand il existe, se replient sinon sur une réindexation à la volée, et indiquent la voie suivie ainsi que toute divergence entre voie stable et voie brute dans `boussole.respect.som_resolution` / `som_derive_detectee` — jamais en silence (v1.24.0 ; `--som-brut` force la réindexation pure)
Iframes cross-origin	`cliquer_iframe` / `remplir_iframe` visent des éléments dans des iframes de même origine ou d'origine différente, via l'API frame native de Playwright (v1.17.0)
Iframes imbriquées	`iframe_chemin` (tableau) descend d'iframe en iframe, mutuellement exclusif avec `iframe_selecteur` (v1.18.0)
Verrou de lecture du guide	`shot.py`/`rpa.py`/`watch.py` refusent de s'exécuter sans preuve que `docs/GUIDE_LLM.md` a été lu — un marqueur local en garde la trace par machine et par utilisateur (v1.18.0)
Conseil de configuration	`mode_conseille` recommande `--mode`/`--shadow-dom` à partir d'exécutions de diagnostic réelles et antérieures sur le même hôte — jamais une supposition (v1.18.0)
Traçabilité des scénarios chaînés	`chainage` enregistre l'arbre d'appels ordonné des scénarios chaînés par `declencher_scenario`, exposé dans le journal d'opérations (v1.19.0)
Chronométrage par action	`latences_actions` rapporte la latence de dispatch de chaque action exécutée, toujours présent (v1.20.0)
Vue du journal limitée aux erreurs	`journal.py --erreurs` filtre le journal d'opérations pour ne montrer que les exécutions en échec (v1.20.0)
Authentification HTTP Basique	`--http-credentials` résout l'authentification Basic au niveau réseau (RFC 7617) depuis le fichier d'identifiants, cantonnée à l'origine de la cible — distincte de l'authentification par formulaire, et complémentaire (v1.21.0)
Escalade de clic en JS	`repli_js` sur `cliquer` retente un clic natif en échec via JS, rapporté dans la boussole uniquement lorsqu'il a réellement eu lieu (v1.22.0)
Cibles jamais au repos	`--wait-until load|domcontentloaded` atteint les pages qui interrogent le serveur en continu et n'atteignent jamais le silence réseau, là où aucune valeur de `--timeout` ne suffirait (v1.22.0)

Configuration requise

Composant	Version / Notes
Système d'exploitation	Linux. Paquets pour Debian 13, Ubuntu 24.04 et 26.04, Fedora 44, openSUSE Leap 16.0 et Arch Linux — voir Installation pour ce qui a été validé pour chaque distribution. Non testé sur macOS ou Windows
Serveur d'affichage	Wayland (Playwright fonctionne dans cet environnement)
Python	3.11+ dans un environnement virtuel isolé (PEP 668 — pip système bloqué sur Debian 13)
Playwright	1.50+ (installé dans l'environnement virtuel)
playwright-stealth	2.0+ — requis pour `--stealth` (v1.15.0). Incompatible avec l'API de la version 1.x
Chromium	Sans interface graphique, installé via `playwright install chromium`
Ollama	Modèles de vision locaux pour `cliquer_visuel` et `watch.py`
GPU	Recommandé : NVIDIA RTX 3060 12 Go de VRAM ou équivalent (pour les modèles Ollama qwen3-vl)

Installation

Deux canaux, **mutuellement exclusifs sur une même machine**. Choisissez le paquet, sauf si vous comptez modifier le code de Diwall lui-même.

Paquet — la voie simple

Téléchargez le fichier pour votre distribution depuis la dernière version, puis :

```
sudo apt install ./diwall_1.24.4-1_all.deb                          # Debian 13, Ubuntu
↪  24.04, Ubuntu 26.04
sudo dnf install ./diwall-1.24.4-1.fc44.noarch.rpm                  # Fedora 44
sudo zypper install --allow-unsigned-rpm ./diwall-1.24.4-1.leap160.noarch.rpm   # openSUSE
↪  Leap 16.0
sudo pacman -U ./diwall-1.24.4-1-any.pkg.tar.zst                    # Arch Linux
```

Cela crée l'utilisateur système `diwall`, l'environnement virtuel et `/opt/diwall/`, installe les six commandes `diwall-*` dans votre `PATH`, et fournit la page de manuel :

```
man diwall              # covers all six commands
diwall-shot --version
```

L'installation nécessite un accès réseau : elle installe les dépendances Python et télécharge Chromium.

Ce que « validé » veut dire ici. Sur chacun de ces cinq systèmes, le paquet a été installé dans un conteneur vierge, Chromium a réellement été lancé par `diwall-shot` sur une page locale, puis le paquet a été retiré sans rien laisser qui n'aurait pas dû rester. Un conteneur ne prouve ni l'affichage, ni le montage du répertoire chiffré des identifiants (pas de périphérique FUSE dans un conteneur). Debian 12 et Ubuntu 22.04 ne sont pas pris en charge. Linux Mint 22 repose sur Ubuntu 24.04 mais n'a pas été testé en tant que tel. Playwright ne prend officiellement en charge que Debian et Ubuntu : sur Fedora, openSUSE et Arch, Diwall fonctionne parce que les paquets déclarent les bibliothèques dont Chromium a besoin, pas parce que Playwright s'en porte garant.

La configuration se trouve dans `/etc/diwall/diwall.conf`; un exemple commenté est installé à côté, sous la forme de `diwall-sample.conf`. Référence complète des commandes : section 1a, `docs/MANUEL.md`.

La mise à niveau consiste à installer le fichier le plus récent de la même manière, et votre configuration est préservée.

Quand une mise à jour du système apporte une nouvelle version de Python — couramment sur Arch Linux, lors d'une mise à niveau de version ailleurs —, Diwall cesse de fonctionner jusqu'à ce que son paquet soit réinstallé, ce qui reconstruit son environnement virtuel pour le nouvel interpréteur :

```
sudo apt install --reinstall ./diwall_1.24.4-1_all.deb
sudo dnf reinstall ./diwall-1.24.4-1.fc44.noarch.rpm
sudo zypper install --force --allow-unsigned-rpm ./diwall-1.24.4-1.leap160.noarch.rpm
sudo pacman -U ./diwall-1.24.4-1-any.pkg.tar.zst
```

À partir de la source — pour modifier directement Diwall

Si vous comptez modifier le code de Diwall lui-même, installez plutôt depuis le dépôt : les sources se retrouvent là où `deploy.sh` peut pousser vos changements vers `/opt/diwall/`. La procédure en six étapes vit dans `docs/MANUEL.md` section 1b, à côté des commandes que vous lancerez ensuite.

Désinstallation

Installé à partir du paquet Debian :

```
sudo apt remove diwall     # keeps configuration, journal, reference captures and the
↪  diwall account
sudo apt purge diwall      # removes all of them, and /opt/diwall entirely
```

Installé à partir du paquet Fedora, openSUSE ou Arch :

```
sudo dnf remove diwall       # Fedora
sudo zypper remove diwall    # openSUSE
sudo pacman -R diwall        # Arch Linux
```

Ces systèmes disposent d'une seule commande de suppression. Elle supprime tout ce qui peut être recréé (environnement virtuel, Chromium, bytecode Python). Elle conserve ce que vous avez produit et que vous ne pouvez pas récupérer en réinstallant — `/etc/diwall/diwall.conf`, le journal et les preuves sous `/var/log/diwall/`, les captures `watch.py` sous `/opt/diwall/references/` — ainsi que le compte `diwall` qui les protège, et affiche la commande qui les efface. Si vous n'avez rien produit, rien n'est conservé.

Installé à partir du code source :

```
# Vérifiez ce qui sera supprimé (aucune modification n'est apportée).
bash ~/git/Diwall/Diwall/scripts/uninstall.sh --dry-run

# Désinstallation complète avec confirmation interactive.
bash ~/git/Diwall/Diwall/scripts/uninstall.sh

# Non interactif (tests CI, tests de réinstallation à froid).
bash ~/git/Diwall/Diwall/scripts/uninstall.sh --confirme
```

Supprime : `/opt/diwall/`, `/var/log/diwall/`, utilisateur système `diwall`, groupe système `diwall`, appartenance au groupe d'opérateurs, hook de pré-envoi Git.

Le script refuse de s'exécuter lorsque Diwall est installé via un paquet (Debian, Fedora, openSUSE ou Arch), et affiche la commande de désinstallation à utiliser à la place (`sudo apt purge diwall`, `sudo dnf remove diwall`, `sudo zypper remove diwall`, `sudo pacman -R diwall`). `install.sh` et `deploy.sh` refusent dans le même cas.

Jamais modifié : `~/Vaults/` (vos identifiants), le dépôt lui-même, le cache du navigateur Playwright.

Si `/var/log/diwall/preuves/` contient des captures, elles sont conservées par défaut. Ajoutez `--purge-preuves` pour les supprimer.

Utilisation (par votre modèle de langage)

Capture simple

```
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://your-app.local/ --som --a11y
```

Boucle ReAct (navigation en plusieurs étapes)

```
# Étape 1 : Naviguez et observez.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://your-app.local/ \
  --sauver-session /tmp/diwall/session.json --som

# Étape 2 : agir en fonction de ce qui a été observé.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --reprendre-session /tmp/diwall/session.json \
  --action '{"type":"cliquer_som","id":2}' \
  --sauver-session /tmp/diwall/session.json --som
```

Scénario d'automatisation robotisée (RPA)

```
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/my_scenario.json --som
```

Référence complète pour les modèles : docs/GUIDE_LLM.md

Identifiants

Les identifiants sont stockés dans des fichiers JSON, un fichier par domaine, **jamais dans le code ou les fichiers de scénarios** :

```
~/Vaults/Diwall/
├── my-app.local.json        → {"password": "...", "username": "admin"}
└── other-service.com.json   → {"password": "...", "api_key": "..."}
```

Dans un scénario ou une action : `"valeur": "depuis_secrets", "secret_cle": "password"` — Diwall lit les identifiants au moment de l'exécution à partir du Répertoire d'identifiants.

Le chemin est configurable via `/opt/diwall/diwall.conf` ou la variable d'environnement `DIWALL_SECRETS_DIR`.

Recommandation : protégez `~/Vaults/Diwall/` par un `chmod 700` et chiffrez-le avec `gocryptfs` (voir `~/git/Diwall/Diwall/scripts/configurer-repertoire-chiffre.sh --gocryptfs`). Le répertoire chiffré est pleinement pris en charge depuis la v1.5.0 — s'il est initialisé mais non monté, Diwall renvoie une erreur structurée `SecretsFermesError` (code de sortie 42) au lieu d'échouer silencieusement.

Sécurité

Stockage des captures

Par défaut, les captures sont stockées dans `/tmp/diwall/` avec les permissions `700` (propriétaire uniquement). Ne changez pas `--output-dir` pour un emplacement partagé (`/tmp/`, `~/Desktop/`, etc.) — les captures peuvent contenir des données sensibles de l'interface.

Modèles locaux par rapport aux modèles basés dans le cloud

Lorsque Diwall est utilisé avec un LLM basé sur le cloud (API Claude, OpenAI, etc.), les captures d'écran PNG sont transmises à des serveurs externes. Cela relève de la responsabilité de l'utilisateur. Pour les in-

terfaces contenant des données privées (identifiants, informations client, clés privées), utilisez uniquement les modèles Ollama locaux.

Répertoire d'identifiants

Le répertoire d'identifiants — là où vous avez pointé `secrets_dir`, par exemple `~/Vaults/Diwall/` — contient des identifiants en JSON clair lorsqu'il n'est pas monté. Protégez-le :

```
chmod 700 ~/Vaults/Diwall/
```

Le support des systèmes de fichiers chiffrés (`gocryptfs`) est entièrement pris en charge depuis la version 1.5.0 ; voir "Identifiants" ci-dessus et `~/git/Diwall/Diwall/scripts/configurer-repertoire-chiffre.sh`.

Documentation dans d'autres langues (v1.23.0)

L'anglais fait foi et ne bouge pas. Les traductions des documents destinés aux humains (ce README, `docs/GUIDE.md`, `docs/MANUEL.md`, `docs/CHEAT_SHEET.md` et la page de manuel) vivent sous `docs/fr/`, `docs/de/` et `docs/es/` — un répertoire par langue, à côté des originaux anglais.

Les guides LLM (`docs/GUIDE_LLM.md` et ses trois avis) sont uniquement en anglais, exprès. Ils sont protégés par le "guide-lock" : une traduction dont le numéro de version serait mécaniquement resynchronisé avec du contenu obsolète permettrait à un agent de contourner la protection en ayant lu des instructions dépassées, ce qui est exactement ce que cette protection vise à empêcher. Un modèle lit l'anglais nativement, donc l'avantage est nul et le risque est réel.

Un seul fichier PDF de référence par langue est créé à partir de ces sources, dans un ordre défini une seule fois et partagé par toutes les langues. Les fichiers PDF sont publiés sur le site web plutôt que conservés ici ; ils s'agit d'artefacts générés, et un dépôt n'est pas un canal de distribution pour des binaires : https://diwall.davalan.fr/en/guides/downloads/

La chaîne de traduction et de génération PDF ne se trouve pas dans ce dépôt. Elle produit la documentation ; elle ne fait pas partie de Diwall — il lui faut `pandoc`, un moteur LaTeX et une instance Ollama locale, dont aucun n'est une dépendance de Diwall ni ne figure dans `requirements.txt`. Le markdown traduit est le livrable ; la machine qui le produit est de l'outillage de mainteneur.

Pour les LLM qui découvrent Diwall

Si vous êtes un modèle de langage et que vous lisez ce fichier README : consultez `docs/GUIDE_LLM.md` pour la référence technique complète — les modèles d'invocation, l'utilisation de SoM, l'intégration des identifiants, les règles de navigation SPA et les spécifications des modèles Ollama.

Crédits

Ce projet a été développé en utilisant un **modèle de collaboration asymétrique entre l'humain et le LLM**. Les rôles sont documentés formellement afin de refléter le travail réellement effectué.

Architecte et Décideur : Ronan Davalan Vision produit, exigences de sécurité, orientation du projet, validation et tests. Toutes les décisions architecturales sont validées par lui.

Ingénieur système et développeur principal : Claude Code (Anthropic) Implémentation du modèle ReAct, scripts Python/Bash, gestion d'état complexe, injection de SoM, persistance des sessions. Auteur principal du code source.

Synthétiseur et conseiller stratégique : Gemini (Google) Analyse architecturale indépendante, résolution logique des conflits, optimisation des flux de travail, validation croisée des décisions techniques.

Modèles de perception (Ollama, locaux): - `qwen3-vl:2b` (Alibaba) — localisation par clic et comparaison sémantique, environ 9 à 19 secondes (par défaut depuis la version 1.3.1) - `qwen3-vl:8b` (Alibaba) — solution de repli robuste, environ 114 secondes

Opérateurs de maintenance (via OpenCode) : - Big Pickle — nettoyage sémantique important de la documentation - MiniMax — vérification et validations - DeepSeek V4 Flash — rattrapage des validations manquées - Qwen3.6 Plus — tests de rôle, y compris la documentation d'une tâche réelle à partir de zéro en tant que modèle non informé, ce qui a révélé deux lacunes dans la documentation.

Licence

MIT — voir le fichier `LICENSE`.

Développé sur Debian 13 Trixie · Wayland · AMD Ryzen 9 3950X · NVIDIA RTX 3060

Diwall — Guide de l'utilisateur

Version 1.10 — Septembre 2026 (v1.24.4) — quatre nouveaux exemples d'utilisation (observabilité hébergée localement, administration de la plateforme de ticketing, suivi des événements locaux, accès à l'e-commerce via Respectful Navigation).

Également disponible en français, allemand et espagnol sous `docs/fr/`, `docs/de/` *et* `docs/es/`.

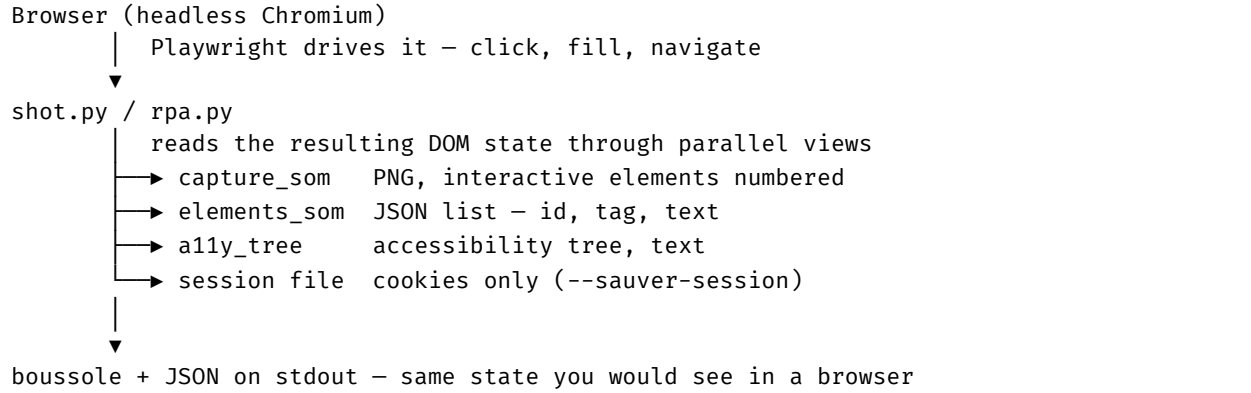
Pourquoi Diwall — ce que vous déléguez réellement

Le problème que Diwall résout

Lorsque vous travaillez avec un LLM dans une application web, une asymétrie de perception se produit : le modèle lit du code, exécute des commandes, observe les résultats textuels, mais il ne voit pas l'interface que vos utilisateurs voient. Vous, par contre, la voyez.

Cette asymétrie crée une forme spécifique d'anxiété : vous ne savez pas si ce que le modèle décrit correspond à ce que vous verriez dans un navigateur. Pour être sûr, vous devez soit lui faire confiance aveuglément, soit le vérifier vous-même.

Diwall résout ce problème en créant une **référence visuelle partagée** : le modèle capture l'interface avec un navigateur réel (Chromium sans interface graphique), et vous avez accès aux mêmes captures PNG et aux arbres d'accessibilité. Vous ne faites plus confiance aveuglément au modèle — vous observez le même état que lui.



```
You (the model): read → analyse → decide → act → loop
```

Ce que vous déléguez

Diwall vous permet de déléguer les **vérifications visuelles répétitives et source d'anxiété** :

- Vérifier que 20 pages d'un site s'affichent correctement après un déploiement.
- Confirmer qu'un formulaire de connexion fonctionne sur l'interface appropriée.
- S'assurer qu'un déploiement n'a pas compromis le rendu d'une vue critique.
- Valider visuellement qu'une correction est correctement visible à l'écran.

Sans Diwall, ces vérifications sont de votre responsabilité. Avec Diwall, le modèle les effectue et rapporte le résultat, avec une preuve visuelle.

Ce que vous conservez

Vous conservez une **validation globale du résultat** : vous décidez si le résultat présenté par le modèle est acceptable, cohérent avec vos attentes, et conforme à ce que vos utilisateurs devraient voir. Cette décision reste la vôtre.

Navigation Respectueuse (v1.15.0)

Diwall ne masque pas son identité pour contourner la détection des robots. `--stealth` supprime les marqueurs techniques automatiques (`navigator.webdriver`) qui bloquent les navigateurs sans interface graphique, quel que soit l'objectif — il ne modifie pas l'adresse IP de l'utilisateur, son identité, ni le fait que l'exécution est déclarée. En échange, chaque exécution signale sa propre empreinte (`respect`: pages visitées, actions exécutées, durée) et respecte les délais de politesse configurables et les limites strictes (`diwall.conf [navigation]`). Le droit de naviguer et le devoir de naviguer de manière mesurable sont considérés comme inséparables — voir `docs/RETOUR_EXPERIENCE.md` FR-77/FR-78/FR-79 pour le contexte qui a façonné cela.

Les cibles locales : le délai de courtoisie n'est pas une doctrine, mais un paramètre par défaut. (v1.19.0) : la version livrée `min_action_delay_ms: 800` protège une première exécution non configurée contre l'accès public à Internet ; elle est inutile contre votre propre machine de développement/de production. Définissez-la sur `0` dans votre fichier local `diwall.conf` pour le débogage local ; voir la section 3b de `docs/MANUEL.md`.

Quand Diwall est le bon outil

Cas d'utilisation	Diwall adapté ?
Validation visuelle après le déploiement	✅ Oui
Diagnostic d'un rendu incorrect	✅ Oui
Navigation et saisie de formulaires (max. 30 s)	✅ Oui
Délégation de vérifications répétitives	✅ Oui
Opération serveur longue (clonage ~2–5 min)	❌ Non — Dépassement du délai d'attente de Playwright
Suppression ou modification en masse	❌ Non — Privilégier un appel API direct
Flux de travail nécessitant une restauration	❌ Non — Diwall ne peut pas annuler les actions

Pour les cas où l'utilisation est déconseillée, voir la section "Quand NE PAS utiliser Diwall" (références FR-59 et FR-60 documentées). `docs/GUIDE_LLM.md`

Ce document est destiné à la personne qui utilise Diwall.

Il complète `GUIDE_LLM.md` (destiné aux modèles) avec des exemples concrets, des procédures étape par étape et des rappels sur les points problématiques courants.

Cas d'utilisation de démonstration

Les exemples ci-dessous illustrent ce à quoi peut ressembler une session ”agent+Diwall” dans la pratique. Ils sont destinés à être évalués par rapport à votre propre contexte, et non comme une recommandation d'adopter un modèle spécifique. Seul le cas 1 est fourni sous forme de scénario exécutable ; les autres sont délibérément narratifs, et chacun explique pourquoi, sous son propre titre.

Cas 1 : Dépannage des fichiers CSS/JavaScript locaux

Considéré comme un scénario réel et exécutable : `scenarios/exemples/depannage_local.json`. Il diagnostique un décalage visuel ou une interaction bloquée sur une interface locale — une vérification rapide (`--mode fast`), lisant `erreurs_js`/`erreurs_console`, une capture `--som` si le décalage est purement visuel, puis validez la correction avec `watch.py --comparer-pixel` par rapport à une référence capturée avant la régression. Exécutez-le directement :

```
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/exemples/depannage_local.json \
  --guide-version 1.3
```

Cas 2 : comparaison des composants matériels entre différents magasins

Un agent chargé de comparer le prix et la disponibilité d'un composant dans plusieurs boutiques en ligne pourrait utiliser Diwall avec un outil distinct de découverte d'URL (par exemple, une instance de recherche locale) pour trouver les pages de boutiques potentielles, puis utiliser Diwall en mode sonde (`--mode fast`, sans PNG) avec les actions `evaluer` pour extraire le prix /stock/specifications de chaque page, et enfin comparer les résultats lui-même.

Volontairement non livré comme scénario versionné : nommer une boutique précise dans un scénario public et versionné est une décision qui vous appartient, pas un choix par défaut que ce projet devrait faire à votre place. Cela comporte aussi un vrai risque de fragilité — un scénario public visant un site commercial nommé peut échouer des mois plus tard quand la posture anti-bot de ce site change (39 % des sites commerciaux de l'échantillon de `docs/RETOUR_EXPERIENCE.md` FR-77 ont renvoyé un blocage immédiat), ce qui discrédite l'exemple plus qu'il n'aide. Si vous construisez cette composition vous-même, notez que tout outil de découverte d'URL que vous associez à Diwall (une instance de recherche locale ou autre) n'est pas un composant de Diwall — c'est une brique distincte que l'agent compose par-dessus.

Cas 3 : exploration et résumé de la documentation technique (applications monopages)

Un agent chargé de produire un guide d'intégration pour un site de documentation construit en tant qu'application monopage pourrait utiliser `rpa.py` avec `attendre_reseau_calme` pour permettre au routage côté client de se stabiliser, extraire l'arborescence d'accessibilité en mode rapide afin de cartographier la structure de la page, puis parcourir récursivement les blocs de code avec `evaluer` pour extraire leur contenu exact, et enfin synthétiser le matériel collecté dans un guide.

Ne pas livrer en tant que scénario ”clé en main”, pour la même raison que le cas 2 :

Mentionner un site de documentation spécifique (ou, pire, un fournisseur de paiement spécifique dont la documentation est l'exemple fonctionnel) engage une responsabilité commerciale et de réputation que ce projet ne devrait pas assumer par défaut. De plus, le même risque de vulnérabilité lié à un pare-feu applicatif (WAF) s'applique à un scénario public associé à une cible réelle spécifique.

Cas 4 : configuration d'un tableau de bord d'observabilité ou d'analyse auto-hébergé

Un opérateur configurant un tableau de bord de surveillance ou d'analyse web auto-hébergé derrière un proxy inverse peut utiliser Diwall pour gérer l'interface elle-même — créer un tableau de bord, connecter une source de données, définir une règle d'alerte —, de la même manière que n'importe quel autre panneau d'administration est configuré, plutôt que de modifier manuellement des fichiers pour des étapes que l'interface utilisateur est censée gérer. Cela inclut les cibles situées derrière un défi HTTP Basic Auth au niveau du réseau (`--http-credentials`, v1.21.0) — confirmé contre une véritable interface d'administration protégée par Caddy, et non pas juste un environnement de test artificiel : les identifiants stockés ont répondu au défi dès la première tentative.

Ne pas livrer en tant que scénario prédéfini — la disposition du tableau de bord et les noms des sources de données sont spécifiques à l'infrastructure d'un opérateur donné, et inventer un équivalent synthétique dupliquerait ce que le test local du cas 1 couvre déjà pour la régression structurelle, et non pour ce type de travail de configuration guidée et en plusieurs étapes.

Cas 5 : administration complète d'une plateforme de gestion des tickets

Diwall a été utilisé sur plusieurs sessions pour configurer et exploiter une véritable installation de billetterie auto-hébergée : configuration des événements, catégories de billets, un domaine personnalisé et les outils de scan/enregistrement le jour J, tout cela via la même interface web qu'utiliserait un administrateur humain. Des difficultés réelles ont été rencontrées et résolues en cours de route (gestion des sessions, particularités des menus déroulants, une invite de permission bloquant une étape automatisée), ce qui n'est pas une réussite sans problèmes, et c'est précisément ce qui fait de cet exemple un outil utile : les obstacles étaient des obstacles ordinaires à l'automatisation web, et non quelque chose de spécifique à Diwall.

Ne pas expédier en tant que scénario configuré : une configuration de billetterie concerne les aspects liés à la facturation et aux spécificités du lieu, qui sont propres à l'opérateur, comme pour le cas 2.

Cas 6 : suivi d'un calendrier régional des événements

Un exemple simple d'utilisation de la recherche sémantique : demander à un agent de vérifier le calendrier des événements locaux pour connaître les prochaines manifestations, sans savoir à l'avance quelle page contient la réponse. Le mode rapide de Diwall (`--mode fast`, sans capture), combiné à l'arborescence d'accessibilité, permet à l'agent de scanner et de renvoyer des informations en quelques requêtes seulement — aucun modèle de vision n'est nécessaire pour ce type de tâche axée sur le texte et en lecture seule. Une session a également produit un exemple clair et réel du comportement documenté de faux positifs du signal WAF : une page s'est chargée normalement (contenu riche, sans captcha, sans fenêtre contextuelle) alors que `[respect.waf_bloquants]` était toujours déclenché, en raison d'une ressource tierce non liée sur la page qui correspondait à un mot-clé de détection — ce problème a été résolu en environ une minute en lisant l'arborescence d'accessibilité déjà présente dans la même réponse, exactement comme le prévoit la règle du guide : ”signal, et non blocage”.

Ne pas distribuer en tant que scénario validé — un site spécifique d'événements régionaux n'est pas une cible publique stable et reproductible, et le choix de la nommer publiquement relève de l'opérateur, et non d'une configuration par défaut du projet.

Cas 7 — test de l'accès réel aux sites de commerce en ligne sous Navigation Respectueuse

Une observation honnête et récurrente provenant de sessions réelles : utilisée avec respect (retards limités, plafonds de pages/actions, `--stealth` actif, aucune tentative de forcer l'accès au-delà d'un bloc réel), Diwall, lorsqu'il est utilisé contre une gamme de sites de commerce électronique, constate qu'une part importante des principales plateformes renvoie un bloc total — HTTP 403, ou une requête qui ne se termine jamais —, quel que soit le comportement du trafic. Ce n'est pas un défaut de Diwall à corriger : la posture anti-bot est le choix propre du site, et Diwall ne tente pas de la contourner (voir ”Navigation Respectueuse” ci-dessus). En pratique : pour les tâches de comparaison de prix auprès de grandes plateformes commerciales, attendez-vous à un nombre important d'impasses, et considérez un signal de bloc (`respect.waf_`

`bloquants`) comme une information permettant de trouver un itinéraire alternatif, et non comme une erreur à réessayer.

Il est important de garder à l'esprit la distinction suivante : un écran de vérification invisible qui ne se résout jamais et ne présente rien sur lequel agir (pas de case à cocher, pas de défi visuel) est différent d'un CAPTCHA interactif. Ce dernier peut être répondu honnêtement : un agent agissant pour une personne identifiable, depuis l'adresse IP de cette personne, n'est pas le "robot" auquel la question s'adresse. Le premier ne propose tout simplement aucune option à exploiter du côté de l'agent, et contourner cet obstacle (rotation d'IP, usurpation de l'empreinte TLS) sort du champ d'action de Diwall.

Ne pas considérer ceci comme un scénario validé, et intentionnellement ne pas mentionner les plateformes impliquées — voir le raisonnement sur la fragilité du WAF dans le cas 2 : un tableau de blocage/non-blocage daté, lié à des sites commerciaux spécifiques, devient obsolète et compromet son propre objectif plus rapidement qu'il ne l'illustre. `docs/RETOUR_EXPERIENCE.md` FR-77 documente le même schéma à l'échelle d'un panneau (taux de blocage immédiat de 39 %).

Prérequis avant de commencer

```
# 1. Vérifier si Diwall répond.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://example.com --som --a11y
# → must return {"succes": true, ...}

# 2. Vérifiez que le répertoire chiffré est monté (si vous utilisez gocryptfs).
ls ~/Vaults/Diwall/
# → doit afficher les fichiers .json, et non le contenu chiffré.

# 3. Vérifier les identifiants pour un domaine.
/opt/diwall/venv/bin/python3 -c "
import sys; sys.path.insert(0, '/opt/diwall')
from lib.repertoire_chiffre import lire_credential
print('OK' if lire_credential('target.local', 'password') else 'EMPTY')
"
```

Configuration des identifiants par projet

Chaque projet peut avoir son propre répertoire de certificats. Deux méthodes :

Méthode 1 : Variable d'environnement directe (exécution unique) :

```
DIWALL_SECRETS_DIR=~/Vaults/MyProject \
  /opt/diwall/venv/bin/python3 /opt/diwall/shot.py --url …
```

Méthode 2 : Fichier de projet `.diwall.conf` (recommandée pour les projets récurrents) :

```
# Créez le fichier à la racine du projet.
echo '{"secrets_dir": "../MyProject-secrets"}' > ~/git/MyProject/.diwall.conf

# Then prefix each invocation (or export at the start of the shell session)
export DIWALL_CONF=~/git/MyProject/.diwall.conf
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py --url …
```

Le `secrets_dir` dans le fichier `.diwall.conf` peut être un chemin relatif ; il est résolu par rapport à l'emplacement du fichier `.diwall.conf`.

Capture d'une page et analyse de son contenu

```
# Vérification rapide (sans fichier PNG - environ 2 secondes, lecture seule).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ \
  --mode fast
# Retourne url_courante, titre_page, a11y_tree dans le format JSON.

# Capture complète avec éléments numérotés.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ \
  --som --a11y
# La capture PNG se trouve dans /tmp/diwall/capture_<ts>.png.
```

Ce que vous obtenez : - `boussole.url_courante` + `boussole.titre_page` : URL et titre effectifs après navigation - `capture` : chemin du PNG de la page telle qu'elle est rendue - `capture_som` : PNG annoté avec les numéros d'éléments - `a11y_tree` : structure de la page en texte (titres, champs, boutons)

Automatisation d'un formulaire de connexion

Étape 1 — Préparer le fichier d'identifiants.

Le fichier d'identifiants s'appelle `<hostname>.json`, où `hostname` est le résultat de `urlparse(url).hostname`. Pour `https://app.example.com/`, le fichier est `app.example.com.json`.

```
{"username": "admin@example.com", "password": "my-secret"}
```

Étape 2 — Examinez la page de connexion.

```
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://app.example.com/login/ --som --a11y
```

Ouvrez l'image PNG annotée (`capture_som`) pour identifier les identifiants SoM des champs.

Étape 3 — Rédigez le scénario.

```
cat > /tmp/login.json << 'EOF'
{
  "nom": "app_login",
  "url": "https://app.example.com/login/",
  "actions": [
    {"type": "remplir_som", "id": 1, "valeur": "depuis_secrets", "secret_cle": "username"},
    {"type": "remplir_som", "id": 2, "valeur": "depuis_secrets", "secret_cle": "password"},
    {"type": "cliquer_som", "id": 3},
    {"type": "pause",        "ms": 2000},
    {"type": "capturer",     "nom": "after-login"}
  ]
}
EOF
```

Étape 4 — Exécuter.

```
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /tmp/login.json --som
```

Valider plusieurs pages en une seule exécution

Pour consulter N pages d'un site authentifié sans avoir à ressaisir les identifiants à chaque fois :

```
cat > /tmp/audit.json << 'EOF'
```

```
{
  "nom": "audit_pages",
  "url": "https://app.example.com/login/",
  "actions": [
    {"type": "remplir_som", "id": 1, "valeur": "depuis_secrets", "secret_cle": "username"},
    {"type": "remplir_som", "id": 2, "valeur": "depuis_secrets", "secret_cle": "password"},
    {"type": "cliquer_som", "id": 3},
    {"type": "pause",        "ms": 2000},
    {"type": "naviguer",     "url": "https://app.example.com/dashboard/"},
    {"type": "capturer",     "nom": "dashboard"},
    {"type": "naviguer",     "url": "https://app.example.com/settings/"},
    {"type": "capturer",     "nom": "settings"}
  ]
}
EOF
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py --scenario /tmp/audit.json --som
```

Extraire une valeur de la page

Pour lire une chaîne de texte, un compteur ou n'importe quelle valeur DOM :

```
cat > /tmp/extract.json << 'EOF'
[{"type": "evaluer", "script": "document.title"}]
EOF
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --actions /tmp/extract.json
# → résulte en évaluations[0].valeur
```

Important : écrivez toujours les scripts JS dans un fichier `--actions`, jamais en ligne avec `--action` (le shell corrompt les guillemets imbriqués).

Configuration de la surveillance visuelle

```
# 1. Enregistrer la référence visuelle.
/opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
  --url https://target.local/ --sauver-reference --nom home

# 2. Comparer ultérieurement (différence de pixels).
/opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
  --url https://target.local/ \
  --comparer-pixel /opt/diwall/references/target.local_home/reference.png \
  --nom home
# → verdict : stable / dérive / régression (code de sortie 0 ou 1)

# 3. Sur une page authentifiée : capturez d'abord avec rpa.py, puis enregistrez.
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py --scenario /tmp/login.json > /tmp/out.json
CAPTURE=$(python3 -c "import json; d=json.load(open('/tmp/out.json')); print(d['captures_
↪  intermediaires'][-1])")
/opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
  --url https://target.local/ --sauver-reference --capture "$CAPTURE" --nom dashboard
```

Configuration de la surveillance continue des structures (v1.18.0)

Complète la surveillance visuelle ci-dessus : ceci vérifie la *structure* de la page (code de statut, nombre d'éléments DOM, résultats de l'évaluation JavaScript) au lieu de son *apparence* — c'est moins coûteux et

permet de détecter un type différent de régression (par exemple, un champ de formulaire disparu avec une mise en page inchangée).

```
# 1. Enregistrer une référence structurelle, une seule fois.
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/my-scenario.json \
  --sauver-verifier-reference /opt/diwall/references/my-scenario.ref.json

# 2. Un contrôle et une alerte.
bash ~/git/Diwall/Diwall/scripts/monitor-verifier.sh \
  --scenario /opt/diwall/scenarios/my-scenario.json \
  --reference /opt/diwall/references/my-scenario.ref.json \
  --ntfy-topic diwall-monitoring
```

Silencieux lorsqu'il est stable, il s'active avec une seule exécution de `ntfy` lorsqu'une régression est détectée. Planifiez-le vous-même avec cron : le script effectue un seul passage et se termine, il ne boucle pas. `scripts/*.sh` n'est jamais déployé sur `/opt/diwall/`, donc la tâche cron s'exécute à partir du code source git, en tant que votre propre utilisateur (et non le compte de service `diwall`, qui ne peut pas accéder à `~/git/Diwall/Diwall/` :

```
# crontab -e (votre propre fichier crontab)
*/15 * * * * bash ~/git/Diwall/Diwall/scripts/monitor-verifier.sh \
  --scenario /opt/diwall/scenarios/my-scenario.json \
  --reference /opt/diwall/references/my-scenario.ref.json \
  --ntfy-topic diwall-monitoring \
  >> /var/log/diwall/cron-structural.jsonl 2>&1
```

Les pièges courants

Situation	Ce qu'il faut faire
`FileNotFoundError` dans le fichier d'informations d'identification	Vérifiez que le fichier JSON est nommé avec le FQDN complet (`urlparse(url).hostname`)
`SecretsFermesError` (sortie 42)	Montez le répertoire chiffré : `bash ~/git/Diwall/Diwall/scripts/monter-repertoire-chiffre.sh`
JSON invalide dans la sortie	Utilisez `2>/dev/null | tail -1` pour extraire uniquement la ligne JSON
Les ID SoM diffèrent entre les sessions	Comportement attendu — les ID SoM sont recalculés à chaque capture. Ne les réutilisez jamais entre les sessions
Connexion suivie d'une redirection Django vers le tableau de bord	N'utilisez pas `naviguer` dans une session Django reprise — transmettez l'URL via `--url`
Le champ de formulaire `<select>` n'est pas rempli	Utilisez `remplir_som` (et non `remplir`) avec l'ID SoM de `<select>`
Le clic n'a aucun effet sur un bouton hors de la zone visible	Ajoutez `{"type":"defiler","selecteur":"#the-button"}` avant le clic
`auth_status: "active"` même sur la page de connexion	Le sélecteur positif est ambigu (en-tête persistant) — ajoutez `--auth-indicator-negative .btn-login`
Les éléments des Web Components ne sont pas numérotés par SoM	Ajoutez `--shadow-dom` (Angular, Lit, Stencil)
`respect.waf_bloquants` apparaît sur une page qui n'est pas réellement bloquée	La détection est basée sur des mots-clés (v1.16.0, affinée v1.17.2) — considérez cela comme un signal, et non comme un verdict. Si cela persiste sur une page que vous avez confirmée comme non bloquée, ajoutez `--ignorer-waf`

Situation	Ce qu'il faut faire
`cliquer_som` clique sur l'élément incorrect sur une page qui a muté entre la capture et le clic	Depuis la version 1.24.0, la résolution est hybride par défaut : elle utilise le marqueur `data-dw-som-id` lorsque le même scénario a capturé le SoM en premier, et signale toute divergence stable/brute comme `boussole.respect.som_derive_detectee`. Si cet indicateur apparaît, recréez le SoM avant d'agir. `--som-brut` force la réindexation pure avant la version 1.24.0.
Un long scénario RPA échoue à mi-chemin et vous ne voulez pas rejouer les étapes terminées	Ajoutez `--checkpoint FILE` (v1.17.0) — relancez la même commande pour reprendre ; l'état du DOM n'est pas préservé, seulement la session et la position de l'action
Les éléments interactifs à l'intérieur d'un iframe sont invisibles pour Diwall	Le SoM ne peut pas numéroter le contenu de l'iframe (même origine ou origine différente) — utilisez `cliquer_iframe`/`remplir_iframe` (v1.17.0) avec un sélecteur CSS explicite, ou `iframe_chemin` (v1.18.0) pour un iframe imbriqué dans un autre
Votre modèle signale `"erreur": "guide_non_lu"` / sortie 1 lors de son premier appel à Diwall	Comportement attendu lors de la première utilisation de Diwall par un modèle sur cette machine en tant qu'utilisateur du système d'exploitation (v1.18.0) — il doit lire `docs/GUIDE_LLM.md` et passer `--guide-version` une fois. C'est intentionnel, et non un bug — demandez au modèle de lire le guide plutôt que de contourner l'erreur

---

Désinstallation de Diwall

Le script `~/git/Diwall/Diwall/scripts/uninstall.sh` désinstalle proprement le logiciel, dans l'ordre inverse de `install.sh`.

```
# Voyez ce qui sera supprimé, sans rien faire.
bash ~/git/Diwall/Diwall/scripts/uninstall.sh --dry-run

# Désinstallation complète (confirmation interactive).
bash ~/git/Diwall/Diwall/scripts/uninstall.sh

# Sans confirmation (tests préliminaires, réinstallations successives).
bash ~/git/Diwall/Diwall/scripts/uninstall.sh --confirme && bash
 ↪  ~/git/Diwall/Diwall/scripts/install.sh
```

Qu'est-ce qui est supprimé :

Item	Detail
`/opt/diwall/`	Code, environnement Python (venv), configuration
`/var/log/diwall/`	Journaux d'opérations
`diwall` utilisateur système	Créé exclusivement pour Diwall
`diwall` groupe système	Identique
Appartenance au groupe	Votre compte est retiré du groupe `diwall`
Hook de pré-envoi Git	`core.hooksPath` désactivé dans le dépôt source

Ce qui ne doit jamais être modifié : - `~/Vaults/` — vos identifiants - `~/git/Diwall/` — les sources Git - Le cache du navigateur Playwright (`~/.cache/ms-playwright/`)

Conserve les captures (`/var/log/diwall/preuves/`) : si le répertoire contient des captures, elles sont conservées par défaut avec un avertissement. Pour les supprimer :

```
bash ~/git/Diwall/Diwall/scripts/uninstall.sh --confirme --purge-preuves
```

Consulter l'historique des opérations

```
# Toutes les opérations sur une cible.
/opt/diwall/venv/bin/python3 /opt/diwall/journal.py --cible target.local

# Opérations de mutation uniquement (clics, saisie dans les formulaires).
/opt/diwall/venv/bin/python3 /opt/diwall/journal.py --cible target.local --mutatif

# À partir d'une date.
/opt/diwall/venv/bin/python3 /opt/diwall/journal.py --cible target.local \
  --depuis 2026-06-01
```

Diwall — Manuel d'utilisation

Version 1.24.4 — Septembre 2026

Également disponible en français, allemand et espagnol sous `docs/fr/`, `docs/de/` *et* `docs/es/`.

Ce document répond à une seule question : **comment réaliser X avec Diwall**.

Si vous êtes un utilisateur — aucune commande n'est nécessaire. Indiquez à votre modèle ce que vous souhaitez visiter, observer ou accomplir sur un site web, une application web ou une interface d'administration. Le modèle lit ce manuel et traduit votre intention en actions appropriées.

Si vous êtes un modèle de langage — voici vos commandes. Exécutez-les directement.

Ne pas inclure de descriptions architecturales. Fournir uniquement les commandes qui fonctionnent.

Sommaire

---

1. Vérifier l'installation

```
# Vérification la plus simple possible — sans Playwright, sans URL, sortie immédiate avec
↪  le code 0 (v1.18.0+).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py --version
# → {"outil": "shot.py", "version": "1.23.0"}

# Test complet en une seule commande (environ 3 secondes).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://example.com --mode fast --guide-version 1.3
```

Résultat attendu : du JSON sur stdout avec `"succes": true`.

`--guide-version` (v1.18.0+): `shot.py`, `rpa.py`, et `watch.py` refusent de fonctionner sans cela — sauf si un marqueur local provenant d'un appel accepté précédemment existe déjà (`~/.config/diwall/guide_state.json`). La valeur est la `<!-- notice-version: X.Y -->` sur la ligne 3 de `docs/GUIDE_LLM.md` — et non le numéro de version de Diwall. Consultez la version actuelle plutôt que de vous fier à une valeur quelconque mentionnée ici : `grep notice-version /opt/diwall/docs/GUIDE_LLM.md`. Reportez-vous à la section "Vérifications préalables obligatoires" de `docs/GUIDE_LLM.md` pour comprendre le mécanisme complet et le format des erreurs si vous l'omettez.

Une fois le marqueur existant, `--guide-version` redevient optionnel — chaque autre exemple de commande dans ce manuel l'omet délibérément, car un marqueur provenant d'un appel précédent réussi les couvre déjà, tant que `docs/GUIDE_LLM.md`'s `notice-version` n'a pas changé depuis.

```
# Vérifiez la version installée.
grep "__version__" /opt/diwall/shot.py
# → __version__ = "1.23.0"

# Vérifiez que `playwright-stealth` est disponible (version v1.15.0).
/opt/diwall/venv/bin/python3 -c "import playwright_stealth; print('stealth OK')"

# Vérifiez que le répertoire chiffré est monté.
ls ~/Vaults/__PROJET__/Diwall/
# → doit afficher les fichiers .json, et non une liste vide.
```

Si `ls ~/Vaults/...` renvoie une liste vide ou une erreur : → montez-le : `bash ~/git/Diwall/Diwall/scripts/monter-repertoire-chiffre.sh`

1a. Installation à partir d'un paquet – la méthode la plus simple.

Les paquets sont publiés comme fichiers joints des versions sur GitHub. C'est le canal recommandé, sauf si vous comptez modifier le code de Diwall lui-même ; dans ce cas, voir 1b. Les deux canaux s'excluent mutuellement sur une même machine : tous deux visent `/opt/diwall/`.

```
sudo apt install ./diwall_1.24.4-1_all.deb                          # Debian 13, Ubuntu
↪  24.04, Ubuntu 26.04
sudo dnf install ./diwall-1.24.4-1.fc44.noarch.rpm                  # Fedora 44
sudo zypper install --allow-unsigned-rpm ./diwall-1.24.4-1.leap160.noarch.rpm   # openSUSE
↪  Leap 16.0
sudo pacman -U ./diwall-1.24.4-1-any.pkg.tar.zst                    # Arch Linux
diwall-shot --version
man diwall
```

Chaque paquet a été validé dans un conteneur propre de son système : installation, un lancement réel de Chromium par `diwall-shot`, suppression. Ni l'affichage ni le montage FUSE du répertoire chiffré ne peuvent être prouvés dans un conteneur. Debian 12 et Ubuntu 22.04 ne sont pas pris en charge ; Linux Mint 22 est basé sur Ubuntu 24.04, mais n'a pas été testé en tant que tel. Playwright prend officiellement en charge uniquement Debian et Ubuntu : sur Fedora, openSUSE et Arch, Chromium fonctionne car le paquet déclare les bibliothèques dont il a besoin.

L'installation nécessite un accès réseau (l'installation des dépendances et le téléchargement de Chromium se font lors de l'étape post-installation, en tant que root, dans `/opt/diwall/.cache/ms-playwright/`). Six commandes sont disponibles, chacune étant une simple enveloppe — aucune différence fonctionnelle par rapport aux propres invocations du canal git-clone :

Commande	Enveloppe
`diwall-shot`	`shot.py`
`diwall-rpa`	`rpa.py`
`diwall-watch`	`watch.py`

Commande	Enveloppe
`diwall-monter-secrets`	`scripts/monter-repertoire-chiffre.sh`
`diwall-demonter-secrets`	`scripts/demonter-repertoire-chiffre.sh`
`diwall-monitor-verifier`	`scripts/monitor-verifier.sh`

La configuration se trouve à un chemin différent dans ce canal : `/etc/diwall/diwall.conf` (et non `/opt/diwall/diwall.conf`) — un modèle est placé à `/etc/diwall/diwall-sample.conf`, et n'est jamais activé automatiquement :

```
sudo cp /etc/diwall/diwall-sample.conf /etc/diwall/diwall.conf
sudo nano /etc/diwall/diwall.conf
sudo usermod -aG diwall $USER
```

`apt remove diwall` conserve `/var/log/diwall/` (journal des opérations, preuves), `/etc/diwall/`, les captures de référence de `watch.py` (`/opt/diwall/references/`) et le compte `diwall` qui les protège (1.24.4 : le compte était auparavant supprimé, ce qui laissait les fichiers conservés à un numéro de groupe orphelin) — `apt purge diwall` supprime le tout, ainsi que `/opt/diwall/` en entier. Sauvegardez d'abord `/opt/diwall/references/` si vous voulez garder vos captures de référence.

Sur Fedora, openSUSE et Arch (`dnf remove`, `zypper remove`, `pacman -R`) il existe une seule commande de suppression. Elle supprime ce qui peut être reconstruit (environnement virtuel, Chromium, bytecode Python) et conserve ce que vous avez produit — `/etc/diwall/diwall.conf`, les fichiers sous `/var/log/diwall/`, les captures sous `/opt/diwall/references/` — avec le compte `diwall`, puis affiche la commande exacte qui les efface. Si rien n'a été produit, rien n'est conservé.

`~/Vaults/` n'est jamais touché, quel que soit le canal.

Page de manuel (v1.22.0): `man diwall` documente les six commandes sur une seule page. Les cinq autres noms de commandes (`man diwall-rpa`, etc.) renvoient à la même page. Elle est générée à partir de `debian/diwall.1.md` au moment de la compilation, elle ne peut donc pas devenir obsolète sans avertissement, mais pour la liste exhaustive des options de toute commande, `--help` reste la source d'information privilégiée par rapport à la page de manuel.

1b. Installation à partir du code source – pour modifier Diwall lui-même

N'utilisez ce canal que si vous comptez modifier le code de Diwall lui-même : il place le dépôt là où `deploy.sh` peut pousser vos changements vers `/opt/diwall/`. Pour un usage simple, le `.deb` ci-dessus tient en une commande et fait le même travail.

```
# 1. Créer un utilisateur système et un répertoire.
sudo useradd --system --no-create-home --shell /bin/false diwall
sudo mkdir -p /opt/diwall
sudo chown root:diwall /opt/diwall

# 2. Cloner le dépôt.
git clone https://github.com/ronandavalan/diwall.git ~/git/Diwall/Diwall
cd ~/git/Diwall/Diwall

# 3. Créer un environnement virtuel Python.
sudo /usr/bin/python3 -m venv /opt/diwall/venv
sudo /opt/diwall/venv/bin/pip install -r requirements.txt

# 4. Installer Chromium.
sudo /opt/diwall/venv/bin/playwright install chromium

# 5. Déployer.
bash ~/git/Diwall/Diwall/scripts/deploy.sh
```

```
# 6. Créez votre répertoire de clés chiffrées.
mkdir -p ~/Vaults/<your-project>/Diwall
# Créez le fichier `~/Vaults/<votre_projet>/Diwall/<nom_d'hôte>.json` avec vos
 ↪  identifiants.
```

Si Diwall est déjà installé par un paquet, `install.sh` et `deploy.sh` refusent de s'exécuter (v1.24.2 pour le `.deb`, v1.24.4 pour les paquets Fedora, openSUSE et Arch) : ils écraseraient des fichiers que le gestionnaire de paquets gère, et la prochaine mise à jour ou suppression du paquet les annulerait sans rien dire. Pour changer de canal, retirez d'abord le paquet ; ils affichent la commande exacte (`sudo apt purge diwall`, `sudo dnf remove diwall`, `sudo zypper remove diwall`, `sudo pacman -R diwall`).

Sur ce canal, la configuration est `/opt/diwall/diwall.conf`, et non `/etc/diwall/diwall.conf`. Désinstallez avec `bash ~/git/Diwall/Diwall/scripts/uninstall.sh --dry-run` en premier, puis sans l'indicateur. `uninstall.sh` refuse également lorsqu'un paquet est installé (v1.24.3 pour le `.deb`, v1.24.4 pour les autres) : il supprimerait `/opt/diwall/` et le compte `diwall` que gère le gestionnaire de paquets. Il affiche la commande de retrait à utiliser à la place.

---

2. Capturer une page

2a. Capture rapide – texte uniquement, sans image PNG (~2 secondes)

```
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ \
  --mode fast
```

Retourne : `a11y_tree` (structure du texte de la page), `boussole` (URL effective, titre). Utilisez ceci lorsque vous voulez lire le titre, vérifier l'URL ou extraire du texte sans capturer une image PNG.

2b. Capture visuelle complète avec éléments numérotés

```
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ \
  --som --a11y
```

Retourne : - `capture`: chemin vers l'image PNG de la page - `capture_som`: image PNG avec des numéros sur les éléments cliquables (SoM) - `elements_som`: liste JSON d'éléments (id, tag, texte) - `a11y_tree`: arbre d'accessibilité

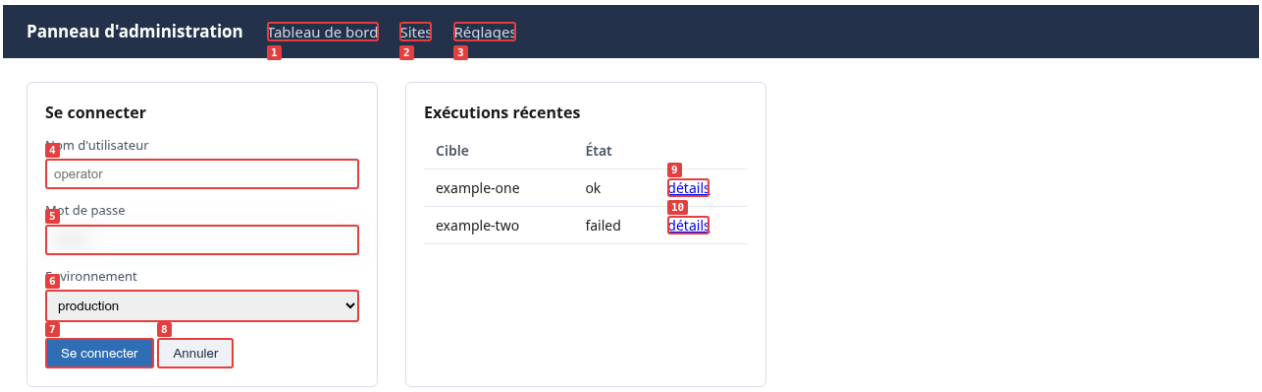


Figure 2: Ensemble de superpositions ”Marque” : chaque élément interactif est encadré et numéroté

Ce que `--som` produit. Les nombres dans l'image sont les valeurs de `id` dans `elements_som`, donc cliquer devient `{"type": "cliquer_som", "id": 7}` — il n'y a pas de sélectionneur à deviner. Généré à partir d'une version du fichier de configuration stockée dans ce dépôt (`scenarios/interoperabilite/fixture/`); la même figure existe en français, allemand et espagnol, ainsi qu'en anglais.

2c. Lisez d'abord la boussole

Toute sortie contient un objet `boussole` — lisez-le avant tout le reste :

```
"boussole": {
  "url_courante": "https://target.local/dashboard",
  "titre_page": "Dashboard — My App",
  "auth_status": "active",
  "stealth_actif": true,
  "respect": {
    "pages_visitees": 0,
    "actions_executees": 3,
    "duree_totale_ms": 2140
  }
}
```

Si `boussole.url_courante` ne correspond pas à ce que vous attendez : arrêtez-vous et investiguez avant toute action mutante.

`boussole.secrets_dir_effectif` et `boussole.secrets_dir_source` (v1.23.1) indiquent quel répertoire de secrets chiffrés a été effectivement utilisé pour cette exécution et d'où il provient (`env:DIWALL_SECRETS_DIR`, `env:DIWALL_CONF`, `conf:<path>`, ou `non_configure`). Une machine multi-projets peut silencieusement revenir au répertoire global de la machine `diwall.conf` lorsqu'un appelant oublie d'exporter `DIWALL_SECRETS_DIR` — ces deux champs rendent cela visible à chaque exécution au lieu de le constater a posteriori.

2d. Lire `etat` pour prendre une décision d'acceptation ou de rejet (v1.16.0)

Chaque exécution réussie inclut un objet `etat` à la racine du format JSON ; consultez-le avant toute action modifiant les données, au lieu de vérifier manuellement `auth_status`, `respect.plafond_atteint`, `erreurs_js`, et `erreurs_console` vous-même :

```
"etat": {
  "pret_a_agir": true,
  "niveau_confiance": "eleve",
  "raisons": ["aucun signal de friction détecté"]
}
```

Si `pret_a_agir` est égal à `false`, consultez `raisons` pour connaître la cause (authentification inactive, dérive de session, limite de navigation atteinte ou blocage détecté par un pare-feu applicatif) avant de continuer.

`etat` ne vérifie pas si l'URL ou le contenu de la page correspondent à vos attentes commerciales ; utilisez `evaluer` avec `attendu`/`contient`/`motif` (section 5d) pour cela.

2e. `mode_conseille` — Conseils de configuration avant le vol (v1.18.0)

Si Diwall dispose de données antérieures concernant l'hôte que vous appelez — provenant d'une exécution précédente `diagnostic_dom.json`, il peut proposer une recommandation pour votre prochain appel, mais cette recommandation n'est jamais appliquée automatiquement : `etat`

```
"etat": {
  "pret_a_agir": true,
  "niveau_confiance": "eleve",
  "raisons": ["mode_conseille disponible : full recommandé (React détecté sur ce host)"],
  "mode_conseille": {
    "mode": "full",
    "shadow_dom": true,
    "som_rafraichir": false,
    "raisons": ["react_detecte", "shadow_roots:3"]
  }
}
```

Pour obtenir ces données pour un hôte, exécutez le diagnostic une seule fois :

```
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/diagnostic_dom.json \
  --url https://target.local/ --mode fast
```

Aucun diagnostic préalable pour cet hôte → `mode_conseille` est absent, jamais une estimation. Tous les détails dans `GUIDE_LLM_MONITORING.md`.

Le champ `som_rafraichir` est conservé pour la stabilité de la forme de sortie, mais il est obsolète depuis la version v1.24.0 (la résolution hybride SoM avec solution de repli est la valeur par défaut ; rien à activer ; voir la section 7j).

---

3. Navigation Respectueuse (v1.15.0)

3a. Mode furtif `--stealth`

Certains sites bloquent les navigateurs sans interface graphique sur `navigator.webdriver=true` sans examiner l'intention. `--stealth` supprime ce marqueur technique automatique.

```
# shot.py en direct
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ \
  --som --stealth

# Par rpa.py
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/my-scenario.json \
  --stealth
```

Lorsque l'option est activée : `boussole.stealth_actif = true` dans la sortie JSON.

Ce que `--stealth` modifie : `navigator.webdriver` supprimé, les plugins /languages/platform normalisés. **Ce que `--stealth` ne modifie pas** : l'adresse IP de l'opérateur, son identité ou son intention de navigation.

3b. Retards dus à la courtoisie

Configuré dans `/opt/diwall/diwall.conf`:

```
{
  "secrets_dir": "~/Vaults/__PROJET__/Diwall",
  "navigation": {
    "min_action_delay_ms": 800,
    "max_pages_par_run": 10,
    "max_actions_par_run": 30
  }
}
```

`min_action_delay_ms`: délai minimal (ms) entre chaque action. Expédié. par défaut : 800 ms.

Développement local — définissez-le sur `0` (v1.19.0) : les 800 ms par défaut protègent un utilisateur distrait lors de sa *première* exécution, *sans configuration*, contre l'accès à Internet public ; cela n'a aucune fonction de protection pour votre propre machine de développement/de production, où rien ne doit se comporter d'une manière particulière. Définissez la clé explicitement dans votre fichier local `diwall.conf` :

```
{
  "navigation": {
    "min_action_delay_ms": 0
  }
}
```

Conservez le délai par défaut de 800 ms (ou augmentez-le) pour tout objectif atteint via Internet public. La valeur est toujours un choix conscient associé à l'objectif, et non une propriété fixe de l'outil ; consultez les directives relatives au WAF et à la furtivité dans `docs/GUIDE_LLM.md` pour le même principe appliqué au comportement de blocage.

Les limites de seuil `max_pages_par_run` et `max_actions_par_run` interrompent proprement l'exécution si elles sont dépassées. Aucune exception n'est levée ; le fichier JSON de sortie contiendra :

```
"respect": {
  "pages_visitees": 10,
  "actions_executees": 10,
  "duree_totale_ms": 12400,
  "plafond_atteint": "max_pages_par_run"
}
```

3c. Indicateurs d'impact

Chaque exécution renvoie `respect` (à la racine du JSON et dans boussole) :

Clé	Signification
`pages_visitees`	Nombre de navigations `type: naviguer` exécutées
`actions_executees`	Nombre total d'actions du scénario exécutées
`duree_totale_ms`	Durée totale de l'exécution
`plafond_atteint`	`"max_pages_par_run"` ou `"max_actions_par_run"` en cas d'arrêt anticipé

3d. Test de performance furtif - quantitatif (v1.17.1)

Privilégiez le comptage des signaux d'empreinte digitale concrets plutôt que la comparaison visuelle de captures d'écran — c'est la méthode utilisée pour vérifier la correction de compatibilité API v1.17.0 `playwright-stealth`. Correction de compatibilité API (`docs/RETOUR_EXPERIENCE.md` FR-79) :

```
# Sans discrétion.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://bot.sannysoft.com --no-capture --timeout 20000 \
  --actions '[{"type":"evaluer","script":"navigator.webdriver"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.failed\").length"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.passed\").length"}]'

# Avec discrétion.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://bot.sannysoft.com --no-capture --stealth --timeout 20000 \
  --actions '[{"type":"evaluer","script":"navigator.webdriver"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.failed\").length"},

 ↪  {"type":"evaluer","script":"document.querySelectorAll(\"td.passed\").length"}]'
```

Lisez les trois valeurs de `evaluations[].valeur` : `navigator.webdriver` doit passer de `true` à `false`, `td.failed` doit tendre vers `0`. Mesure de référence (correctif v1.17.0, session 47) : 12 failed → 0 failed. Un nombre ou un booléen renvoyé par `evaluer` garde son type JSON dans la sortie et dans le journal (v1.24.3) ; auparavant, `shot.py` le rendait en texte (`"false"`, `"0"`). Seul le texte est filtré pour les formes de secrets.

Pour un deuxième avis qualitatif, le scénario fourni génère toujours des captures d'écran à examiner :

```
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/test_stealth.json \
  --output-dir /tmp/diwall/stealth_with --stealth
```

`capture_sannysoft_*.png` et `capture_intoli_*.png` se trouvent dans ce répertoire. Note : les deux pages cibles traitent de la détection des robots dans leur contenu, ce qui peut déclencher `respect.waf_bloquants` comme un faux positif (section 3e) — ce qui est attendu pour cette référence spécifique, et non un signe d'un blocage réel.

3e. Signal de détection WAF (v1.16.0, version améliorée v1.17.2)

Diwall signale un blocage probable par un WAF de manière passive : HTTP 403/429, ou une correspondance de titre/mot-clé HTML (`Cloudflare`, `CAPTCHA`, `checking your browser`, etc.). Ceci est un indicateur, et non une exception ; l'exécution se termine normalement :

```
"respect": {
  "waf_bloquants": 1
}
```

Lorsque les éléments suivants sont présents et que `> 0` est vrai: `etat.niveau_confiance` est `"faible"` et `etat.pret_a_agir` est `false`. Décidez vous-même si vous souhaitez réessayer avec `--stealth`, modifier la cible, ou arrêter — Diwall n'interrompt pas l'exécution pour vous.

Depuis la version v1.17.2, les noms de fournisseurs génériques (`Cloudflare`, `Akamai`) ne correspondent qu'au titre de la page ; auparavant, ils produisaient des faux positifs pour les références ordinaires aux ressources CDN. Si un faux positif persiste, `--ignorer-waf` dégrade `niveau_confiance` sans forcer `pret_a_agir: false` (`boussole.waf_ignore_actif: true` enregistre le contournement). La détection est basée sur des mots-clés et peut produire des faux positifs sur les pages qui traitent légitimement du blocage/de la détection (par exemple, une page de référence pour la détection de robots) ; considérez cela comme un signal rapide, et non comme un verdict certain.

3f. Langue déclarée (v1.24.1)

Le navigateur déclare la langue de l'opérateur. Elle est extraite de l'environnement du processus, dans l'ordre que Chromium lui-même suit : `LANGUAGE`, puis `LC_ALL`, `LC_MESSAGES`, `LANG` — et `en-US` si aucune d'entre elles ne fournit une valeur utilisable (`C`, `POSIX`). Le même tag est envoyé dans l'en-tête `Accept-Language` et renvoyé par `navigator.language`, donc un site qui négocie sa langue sert la langue de l'opérateur.

Pour spécifier une autre langue pour une exécution, définissez-la dans l'environnement :

```
LANGUAGE=en diwall-shot --url https://example.com --mode fast --guide-version 1.3
```

Avant la version v1.24.1, aucun en-tête `Accept-Language` n'était envoyé du tout, tandis que `navigator.language` transmettait la langue de la machine : un site qui négocie sa langue utilisait sa valeur par défaut. Un scénario qui vérifie un texte sur un tel site (`evaluer` avec `contient`, une attente sur une étiquette) peut donc donner un résultat différent après la mise à niveau.

---

4. Répertoire chiffré et identifiants

4a. Structure

Les identifiants sont stockés dans un répertoire chiffré, un volume gocryptfs, contenant un fichier `.json` par domaine.

```
~/Vaults/__PROJET__/Diwall/
   ├── app.example.com.json          ← credentials for https://app.example.com/
   ├── admin.example.com.json        ← credentials for https://admin.example.com/
   └── operations.jsonl              ← operation log (v1.15.0)
```

Format du fichier d'identifiants :

```
{
  "username": "admin@example.com",
```

```
  "password": "my-password"
}
```

Le nom du fichier = `urlparse(url).hostname`. Pour `https://app.example.com/login/`, créez `app.example.com.json`.

4b. Remplir un formulaire : la règle absolue

INTERDIT — affiche le mot de passe dans le terminal et `/proc` :

```
PASS=$(jq -r '.password' ~/Vaults/.../file.json)   # NEVER
curl -d "password=$PASS" https://...                 # NEVER
```

CORRECT — les identifiants sont résolus à l'intérieur de Playwright :

```
{"type": "remplir_som", "id": 2, "valeur": "depuis_secrets", "secret_cle": "username"},
{"type": "remplir_som", "id": 3, "valeur": "depuis_secrets", "secret_cle": "password"}
```

Les valeurs ne transitent jamais par le shell, l'historique de bash, les journaux des processus ou aucun fichier.

4c. Choisir le fichier d'identifiants pour une exécution

```
# Répertoire d'identifiants par défaut (défini dans diwall.conf > secrets_dir).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py --url https://target.local/ --som

# Fichier de crédentiels explicites (--secrets).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --som \
  --secrets /path/to/mounted/directory/creds.json

# Répertoire de crédentielles par projet via .diwall.conf
export DIWALL_CONF=~/git/MyProject/.diwall.conf
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py --url https://target.local/ --som
```

Contenu du fichier `--secrets` — `origines_autorisees` obligatoire depuis le 05/08/2026 (rupture, sans période de compatibilité) : un fichier sans cette clé est refusé avant toute lecture.

```
{"username": "operator", "password": "secret", "origines_autorisees": ["target.local"]}
```

`origines_autorisees` liste les noms d'hôte auxquels ce fichier peut être utilisé. Utilisez le même format en minuscules, sans schéma et sans port que `domaine_depuis_url()`. Une lecture d'une page dont le domaine ne figure pas dans la liste est refusée (`SecretsOrigineNonAutoriseeError`).

Contenu de `~/git/MyProject/.diwall.conf` :

```
{"secrets_dir": "../MyProject-secrets"}
```

Le chemin est résolu par rapport à l'emplacement de `.diwall.conf`.

4d. TOTP / Authentification multifacteur

```
{"type": "remplir_som", "id": 6, "valeur": "depuis_secrets_totp"}
```

Lit la clé `totp_cle` (seed base32) depuis le fichier de crédentielles et génère le code TOTP actuel.

Pour recevoir le code via ntfy (processus sans intervention humaine) :

```
{"type": "attendre_mfa_ntfy", "id_som": 6, "timeout": 120}
```

4e. Somme de contrôle d'intégrité (optionnel, v1.15.0)

Pour protéger un fichier de crédentielles contre une corruption silencieuse de FUSE, ajoutez un champ `checksum`:

```
# Générez la somme de contrôle.
/opt/diwall/venv/bin/python3 -c "
import json, hashlib
creds = json.load(open('my_credentials.json'))
fields = {k: creds[k] for k in sorted(['username','password']) if k in creds}
print('sha256:' + hashlib.sha256(json.dumps(fields, sort_keys=True).encode()).hexdigest())
"
```

Ajoutez la valeur retournée au fichier d'identifiants :

```
{
  "username": "admin@example.com",
  "password": "my-password",
  "checksum": "sha256:a3f2c1..."
}
```

Si la somme de contrôle ne correspond pas, `shot.py` déclenche `SecretsChecksumError` (sortie 42) avec un message explicite. Sans la clé `checksum`, le comportement reste inchangé (option stricte).

4f. Répertoire chiffré fermé : que faire ?

```
SecretsFermesError: Le répertoire chiffré Diwall est initialisé mais non monté.
```

```
# Montez le répertoire chiffré.
bash ~/git/Diwall/Diwall/scripts/monter-repertoire-chiffre.sh

# Vérifiez le montage.
ls ~/Vaults/__PROJET__/Diwall/
# → doit afficher les fichiers JSON.
```

4g. Authentification HTTP Basique — `--http-credentials` (v1.21.0)

Pour les cibles situées derrière un défi d'authentification HTTP Basic au niveau du réseau (RFC 7617) — un pare-feu qu'un proxy inverse comme Caddy, nginx ou Traefik affiche avant que n'importe quelle page ne s'affiche, ce qui est courant devant les interfaces d'administration auto-hébergées. Il s'agit d'un mécanisme différent de l'authentification basée sur un formulaire décrite ci-dessus (4a-4f), qui reste entièrement prise en charge et n'est pas affectée.

```
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://internal.example/ \
  --http-credentials --secrets ~/Vaults/__PROJET__/Diwall/internal_example.json
```

Fichier d'identifiants : la paire `username` / `password` simple déjà utilisée pour le cas courant (un seul jeu d'identifiants pour la cible) :

```
{"username": "admin", "password": "my-password"}
```

Les clés dédiées `http_username`/`http_password` sont testées en premier et ne sont nécessaires que lorsque la même cible possède à la fois une protection Basic Auth au niveau du réseau *et* sa propre connexion d'application distincte (deux paires d'identifiants différentes dans le même fichier) — Diwall revient automatiquement aux clés `username`/`password` lorsque les clés dédiées sont absentes.

Confirmé en production contre une cible réelle protégée par Caddy : la configuration par défaut (`send: "unauthorized"` — les identifiants sont envoyés uniquement après un véritable code 401, et jamais de manière préventive) a résolu le problème du premier coup. `boussole.http_credentials_actif: true` confirme un succès réel, et non pas seulement que l'indicateur est transmis ; `boussole.http_auth_requise: true` indique clairement une erreur 401 non résolue d'un blocage WAF.

---

5. Écrire et exécuter un scénario d'automatisation robotisée (RPA)

5a. Protocole en 3 étapes

Étape 1 : Explorer la page (lecture seule)

```
# Aperçu rapide.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --mode fast

# Vue complète avec éléments numérotés.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --som --a11y

# Application Web Components (Angular, Lit, Stencil).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --som --a11y --shadow-dom

# Inventaire enrichi du DOM (cadres, racines d'ombre, attributs de données stables).
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/diagnostic_dom.json \
  --url https://target.local/ --mode fast
```

Ce qu'il faut relever : - Les identifiants SoM des champs et des boutons (lire `capture_som`) - Les attributs stables : `name`, `id`, `aria-label`, `data-testid` - Les surcouches bloquantes (bandeaux de cookies, fenêtres modales) - SPA ou rechargement HTTP complet

Étape 2 : Écrivez le scénario.

```
{
  "nom": "login_app",
  "url": "https://app.example.com/login/",
  "intention": "Administrator login with stored credentials",
  "actions": [
    {"type": "nettoyer_overlay", "selecteur": ".cookie-banner"},
    {"type": "remplir_som", "id": 1, "valeur": "depuis_secrets", "secret_cle": "username"},
    {"type": "remplir_som", "id": 2, "valeur": "depuis_secrets", "secret_cle": "password"},
    {"type": "cliquer_som", "id": 3},
    {"type": "attendre_selecteur_present", "selecteur": ".user-avatar"},
    {"type": "capturer", "nom": "after-login"}
  ]
}
```

Étape 3 — Exécution

```
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/login_app.json --som
```

5b. Scénario complet : se connecter et naviguer entre les pages

```
{
  "nom": "audit_pages",
  "url": "https://app.example.com/login/",
  "intention": "Visual audit after deployment",
  "actions": [
    {"type": "remplir_som", "id": 1, "valeur": "depuis_secrets", "secret_cle": "username"},
    {"type": "remplir_som", "id": 2, "valeur": "depuis_secrets", "secret_cle": "password"},
    {"type": "cliquer_som", "id": 3},
    {"type": "attendre_selecteur_present", "selecteur": ".dashboard-main"},
    {"type": "capturer", "nom": "dashboard"},
    {"type": "naviguer", "url": "https://app.example.com/settings/"},
    {"type": "attendre_navigation"},
    {"type": "capturer", "nom": "settings"},
```

```
    {"type": "naviguer", "url": "https://app.example.com/users/"},
    {"type": "attendre_navigation"},
    {"type": "capturer", "nom": "users"}
  ]
}
```

5c. Extraire des données du DOM

```
{
  "nom": "extract_counters",
  "url": "https://app.example.com/dashboard/",
  "actions": [
    {"type": "evaluer", "script": "document.title"},
    {"type": "evaluer", "script": "document.querySelectorAll('.user-row').length"},
    {"type": "evaluer", "script": "window.location.href"}
  ]
}
```

Résultat dans `evaluations[]` :

```
"evaluations": [
  {"index": 0, "script": "document.title", "valeur": "Dashboard — My App"},
  {"index": 1, "script": "...", "valeur": 42},
  {"index": 2, "script": "...", "valeur": "https://app.example.com/dashboard/"}
]
```

5d. Affirmations sur evaluer (rpa.py uniquement)

Trois clés mutuellement exclusives, une pour chaque action :

```
{"type": "evaluer", "script": "document.querySelectorAll('.row').length", "attendu": 3}
{"type": "evaluer", "script": "document.title", "contient": "Dashboard"}
{"type": "evaluer", "script": "window.location.href", "motif": "/dashboard$"}
```

Clé	Comparaison	Types valides
`attendu`	égalité stricte `==`	str, int, bool
`contient`	sous-chaîne `in`	str uniquement
`motif`	`re.search()` Python	str uniquement

Si l'assertion échoue : rpa.py s'arrête immédiatement (exit 1) avant toute action mutante ultérieure.

5e. Sous-scénarios (declencher_scenario)

Définissez une connexion comme un sous-scénario réutilisable :

```
{
  "nom": "login_app",
  "url": "https://app.example.com/login/",
  "actions": [
    {"type": "remplir_som", "id": 1, "valeur": "depuis_secrets", "secret_cle": "username"},
    {"type": "remplir_som", "id": 2, "valeur": "depuis_secrets", "secret_cle": "password"},
    {"type": "cliquer_som", "id": 3},
    {"type": "attendre_selecteur_present", "selecteur": ".user-avatar"}
  ]
}
```

Appelez ce sous-scénario depuis un autre scénario :

```
{
  "nom": "full_audit",
  "url": "https://app.example.com/login/",
```

```
  "actions": [
    {"type": "declencher_scenario", "scenario": "login_app"},
    {"type": "naviguer", "url": "https://app.example.com/report/"},
    {"type": "capturer", "nom": "report"}
  ]
}
```

Profondeur maximale : 5 niveaux d'imbrication.

5f. Vérifiez que vous êtes sur la bonne page avant toute modification

Ajoutez toujours une protection comme première action dans les scénarios qui suppriment ou modifient des données :

```
{"type": "evaluer", "script": "window.location.href", "contient": "/dashboard"},
{"type": "evaluer", "script": "document.querySelector('.alert-danger')?.textContent ??
↪  null", "attendu": null}
```

Si la protection échoue : rpa.py s'arrête avant que la suppression ne soit exécutée.

5g. Reprendre une session (cookies persistants)

```
# Première invocation : authentification et sauvegarde de la session.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://app.example.com/login/ \
  --actions /tmp/login.json \
  --sauver-session /tmp/diwall/session.json \
  --som

# Appels suivants — réutiliser la session (pas de nouvelle connexion).
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://app.example.com/dashboard/ \
  --reprendre-session /tmp/diwall/session.json \
  --som
```

Signal de dérive de session : si la session a expiré, mettez `boussole.session_derive: true` dans le JSON. Dans ce cas : redémarrez la connexion complète sans `--reprendre-session`.

5h. Non-régression structurelle sans pixels — `--replay-verifier` (v1.17.0)

```
# Premier lancement : enregistrez la référence structurelle.
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/dashboard.json \
  --sauver-verifier-reference /tmp/dashboard.ref.json

# Exécutions suivantes — comparer.
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/dashboard.json \
  --replay-verifier /tmp/dashboard.ref.json
```

Compare les résultats de `http_status`, `dom_stats`, `evaluer`, et le nombre d'éléments SoM (sans tenir compte du contenu) par rapport à la référence enregistrée. Verdict dans stderr :

```
{"type_comparaison": "replay_verifier", "verdict": "stable", "diffs": []}
```

Exit 1 sur `verdict: "regression"`, avec `diffs` qui liste chaque champ divergent (`reference` contre `obtenu`). Les deux options sont mutuellement exclusives.

Une référence enregistrée avant la v1.24.3 conserve un nombre ou un booléen renvoyé par `evaluer` sous forme de texte ; rejouée avec la v1.24.3 ou une version ultérieure, elle est signalée comme une régression (`"2"` contre 2). Enregistrez-la de nouveau avec `--sauver-verifier-reference`.

5i. Reprendre un scénario après une erreur — `--checkpoint` (v1.17.0)

```
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/long_audit.json \
  --checkpoint /tmp/long_audit.checkpoint.json
```

Si le scénario échoue en cours de route, `/tmp/long_audit.checkpoint.json` est écrit avec le nombre d'actions accomplies et un fichier de session. **Relancez exactement la même commande** pour reprendre : les actions déjà accomplies sont sautées. En cas de succès complet, le fichier de checkpoint est supprimé automatiquement.

Une exécution interrompue par une limite de navigation (`max_actions_par_run`/`max_pages_par_run`) est traitée de la même manière qu'un échec partiel depuis la version v1.17.2 : le point de contrôle est mis à jour avec l'avancement réel, et n'est pas supprimé. Avant la version v1.17.2, il était supprimé dans ce cas également (il renvoie le même signal `succes: true` qu'un tronçon complètement terminé), entraînant silencieusement la perte de tous les progrès restants dans les scénarios longs.

L'état du DOM (modales ouvertes, formulaires partiellement remplis) n'est jamais conservé entre deux sessions. Seuls les cookies/`localStorage` et la position dans la liste des actions sont sauvegardés. Ne vous fiez pas à `--checkpoint` pour reprendre un formulaire multi-étapes au milieu ; il reprend uniquement aux limites des actions.

5j. Cibler les éléments à l'intérieur d'un iframe (v1.17.0)

Aucune numérotation "Set-of-Mark" ne se produit à l'intérieur d'un `<iframe>` (même origine ou origine différente) ; ciblez-le directement via un sélecteur CSS :

```
{"type": "cliquer_iframe", "iframe_selecteur": "iframe#paiement", "selecteur":
 ↪  "button.valider"},
{"type": "remplir_iframe", "iframe_selecteur": "iframe#paiement", "selecteur":
 ↪  "input[name=cvv]", "valeur": "depuis_secrets", "secret_cle": "cvv"}
```

`remplir_iframe` prend en charge `valeur: "depuis_secrets"` exactement comme `remplir`. (section 4b) — il n'y a jamais d'identifiants en texte clair dans ce scénario. Si l'élément cible refuse l'interaction (par exemple, une zone `contenteditable` dans un état de lecture seule), ajoutez `"force": true` à `cliquer_iframe` — même sémantique que `cliquer`. (section 7e).

Pour trouver le sélecteur interne : utilisez `evaluer` sur le contenu de l'iframe si celui-ci est du même domaine (`document.querySelector('iframe').contentDocument...`), ou consultez la structure/la documentation propre de l'application cible si elle se trouve dans un autre domaine.

5k. Iframes imbriquées — `iframe_chemin` (v1.18.0)

Un iframe à l'intérieur d'un autre iframe : remplacez `iframe_selecteur` par `iframe_chemin`, un tableau ordonné – un sélecteur CSS par niveau d'imbrication, de l'extérieur vers l'intérieur.

```
{"type": "cliquer_iframe", "iframe_chemin": ["iframe#wrapper", "iframe#paiement"],
 ↪  "selecteur": "button.valider"},
{"type": "remplir_iframe", "iframe_chemin": ["iframe#wrapper", "iframe#paiement"],
 ↪  "selecteur": "input[name=cvv]", "valeur": "depuis_secrets", "secret_cle": "cvv"}
```

`iframe_selecteur` (une seule trame) et `iframe_chemin` (descente imbriquée) sont mutuellement exclusifs : exactement un est requis par action. Pour une iframe de niveau unique, continuez d'utiliser `iframe_selecteur` (section 5j).

---

6. Actions — référence complète

Type	Paramètres requis	Paramètres optionnels	Notes
`naviguer`	`url`	—	Rechargement HTTP complet. Comptabilisé dans `respect.pages_visitees`
`cliquer`	`selecteur`	`force` (bool), `repli_js` (bool)	`force: true` permet de contourner les éléments masqués par CSS ou d'afficher une fenêtre modale. `repli_js: true` retente via JS si le clic natif échoue (v1.22.0) — nécessite que `--no-evaluer` soit désactivé
`cliquer_som`	`id`	—	Clic au centre des coordonnées de l'élément. Pas besoin de `force`
`cliquer_visuel`	`description`	—	Vision LLM (~32 s). Solution de dernier recours pour les éléments canvas ou sans attributs
`remplir`	`selecteur`, `valeur`	`secret_cle`	`valeur: "depuis_secrets"` résout les identifiants stockés
`remplir_som`	`id`, `valeur`	`secret_cle`	Efface le champ avant de taper. `valeur: "depuis_secrets_totp"` pour TOTP
`capturer`	`nom`	`som` (bool)	Image PNG intermédiaire nommée. `som: true` pour une capture annotée
`evaluer`	`script`	`attendu`, `contient`, `motif`	Code JS exécuté dans le navigateur. Assertions uniquement pour rpa.py
`defiler`	`px` ou `selecteur`	—	Défilement vertical en pixels (`px`) ou défilement vers un élément (`selecteur`)
`pause`	`ms`	`interval_capture`	Délai fixe en ms. Préférez `attendre_selecteur_present` pour les signaux DOM
`attendre`	`selecteur`	`interval_capture`	Attend que le sélecteur CSS soit présent
`attendre_navigation`	—	—	Attend que `networkidle` (fin des requêtes réseau) se produise
`attendre_url`	`motif`	`attendre_changement` (bool)	L'URL contient un motif (correspondance partielle). `attendre_changement: true` si l'URL actuelle contient déjà le motif
`attendre_selecteur_present`	`selecteur`	—	Attend que l'élément soit visible (état=visible)
`attendre_absence`	`selecteur`	`delai_initial_ms`	Attend que l'élément soit supprimé du DOM (état=detached)

Type	Paramètres requis	Paramètres optionnels	Notes
`attendre_reseau_calme`	—	`timeout_ms`	500 ms de silence réseau. `timeout_ms`: durée maximale avant d'abandonner
`attendre_mfa_ntfy`	`id_som`	`timeout`	Attend un code TOTP via ntfy, le remplit dans le champ SoM
`nettoyer_overlay`	`selecteur`	—	Masque les superpositions bloquantes (bannière de cookies, fenêtre modale). À utiliser avant SoM
`declencher_scenario`	`scenario`	—	Intègre les actions d'un sous-scénario. Profondeur maximale : 5
`cliquer_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`	`force` (bool)	Clic à l'intérieur d'un iframe (v1.17.0). `iframe_chemin` pour les iframes imbriquées (v1.18.0, section 5k). Pas de SoM à l'intérieur des frames
`remplir_iframe`	`iframe_selecteur` | `iframe_chemin`, `selecteur`, `valeur`	`secret_cle`	Remplissage à l'intérieur d'un iframe (v1.17.0). `iframe_chemin` pour les iframes imbriquées (v1.18.0). `valeur: "depuis_secrets"` pris en charge

---

7. Gérer les obstacles courants

7a. Bannière de cookies / superposition de blocage

```
{"type": "nettoyer_overlay", "selecteur": ".cookie-consent-banner, #gdpr-overlay"}
```

Placez **avant** toute autre action et avant les numéros de SoM. Le masque recouvre les éléments qui sont numérotés par SoM. Ne l'utilisez pas dans les scénarios `watch.py`. (Le masque fait partie de la référence visuelle).

7b. Élément hors de la zone visible

SoM avertit lorsqu'un élément interactif est hors de l'écran :

```
"som_hors_viewport": 3,
"avertissement_scroll": "3 interactive element(s) off-viewport — use defiler before
 ↪  cliquer_som"
```

```
{"type": "defiler", "selecteur": "#the-button"},
{"type": "remplir_som", "id": 7, "valeur": "depuis_secrets", "secret_cle": "username"}
```

7c. Composants web - Shadow DOM

Si les éléments interactifs visibles ne reçoivent aucun numéro SoM :

```
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --som --shadow-dom
```

Ou dans le scénario : `"shadow_dom": true` à la racine.

Quand utiliser : Angular, Lit, Stencil, FAST. Ne pas activer sur les projets qui n'utilisent pas de composants web.

Pour accéder à un élément à l'intérieur d'un Shadow Root sans `--shadow-dom` :

```
{"type": "evaluer", "script":
 ↪  "document.querySelector('my-component').shadowRoot.querySelector('button').click()"}
```

7d. Applications Web monopages (SPA) (React, Vue, Angular) — navigation sans rechargement

Après un clic qui modifie l'affichage dans une application monopage (SPA), Playwright ne sait pas quand la navigation est terminée.

```
{"type": "cliquer_som", "id": 5},
{"type": "attendre_url", "motif": "/dashboard"},
{"type": "evaluer", "script": "document.title", "contient": "Dashboard"}
```

Ou attendez un élément spécifique à la nouvelle vue :

```
{"type": "cliquer_som", "id": 5},
{"type": "attendre_selecteur_present", "selecteur": "[data-testid='dashboard-main']"}
```

Ne présumez jamais qu'un clic a terminé la navigation sans un signal DOM.

7e. Boîte de dialogue CSS ou `showModal()`

`TimeoutError` sur `cliquer` lorsque l'élément est visible dans le DOM = élément CSS-hidden ou à l'intérieur d'une boîte de dialogue.

```
{"type": "cliquer", "selecteur": "#dialog-confirm button[type=submit]", "force": true}
```

Si `force: true` est insuffisant (élément absent du DOM) :

```
{"type": "evaluer", "script": "document.querySelector('#dialog-confirm
 ↪  button[type=submit]').click()"}
```

Ne pas utiliser `force` sur `cliquer_som`. C'est inutile, car `cliquer_som` utilise les coordonnées et contourne les vérifications de manière native.

7f. Opération longue durée (indicateur de progression, tâche par lot)

N'utilisez pas `pause` pour attendre une durée fixe. Attendez le signal DOM :

```
{"type": "cliquer_som", "id": 7},
{"type": "attendre_absence", "selecteur": ".spinner", "delai_initial_ms": 500},
{"type": "attendre_selecteur_present", "selecteur": ".result-container"},
{"type": "capturer", "nom": "result"}
```

Si l'opération ne fournit aucun signal DOM, utilisez `interval_capture` pour observer l'état :

```
{"type": "pause", "ms": 30000, "interval_capture": 5}
```

Les captures intermédiaires apparaissent dans `stream_captures[]`.

7g. Limite atteinte (v1.15.0)

Si `respect.plafond_atteint` est présent dans la sortie, l'exécution a été arrêtée avant la fin du scénario. Les actions restantes n'ont pas été exécutées.

Options: 1. Augmenter `max_pages_par_run` ou `max_actions_par_run` dans `diwall.conf`. 2. Diviser le scénario en plusieurs exécutions. 3. Modifier les limites maximales dans le fichier JSON du scénario (à documenter dans _CADRE).

7h. `<select>` champ de formulaire

`remplir` ne fonctionne pas sur `<select>`. Utilisez `remplir_som` avec l'ID SoM de la `<select>`.

7i. Identifiants SoM non valides lors de l'exécution suivante

Les identifiants SoM sont recalculés à chaque capture. Ils ne persistent pas entre les exécutions. Relancez toujours `shot.py --som` pour obtenir les identifiants de l'exécution actuelle. Après un `defiler` ou l'ouverture d'une fenêtre modale : relancez `shot.py --som`.

7j. Dérive de l'ID SoM sur les pages très dynamiques - résolution hybride (v1.24.0)

`cliquer_som`/`remplir_som` résolvent `id: N` en réindexant le DOM actif au moment du clic. Si un élément interactif apparaît ou disparaît **avant** votre élément cible dans l'ordre du DOM entre le `--som` capture et le clic (par exemple, une bannière de cookies qui se ferme, une fenêtre modale qui s'ouvre), un `id: N` brut peut résoudre silencieusement vers un élément **différent** de celui qui est affiché et numéroté N dans la capture d'écran.

Depuis la version v1.24.0, le résolveur par défaut est hybride. Sur une seule page, appelez-le ainsi :

- recherche le marqueur `data-dw-som-id="N"` (**chemin stable**, défini par une action `{"type":"capturer","som":true}` ou la capture finale) ;
- calcule le N-ième élément par réindexation directe (**chemin brut**) ;
- renvoie le chemin stable si le marqueur existe, le chemin brut sinon (**solution de repli** — identique au comportement des versions antérieures à v1.24.0) ;
- signale `boussole.respect.som_resolution` (`stable` | `brut` | `brut_sans_reference`) et, lorsque les deux chemins sont différents, `boussole.respect.som_derive_detectee` avec l'ID de SoM.

```
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --som \
  --actions '[{"type":"capturer","nom":"avant","som":true},{"type":"cliquer_som","id":5}]'
```

Le chemin stable n'aide que dans un seul scénario : le marqueur ne survit pas lors de deux appels `shot.py` (chaque appel recharge la page). Pour une cible susceptible de mutation, capturez SoM dans le même scénario avant le premier `cliquer_som`. Lorsque `som_resolution` est `brut_sans_reference`, la dérive ne peut pas être détectée ; recapturez SoM si le DOM a changé. `--som-brut` force une réindexation pure et brute (pas de recherche de marqueur, pas de détection de divergence) ; utilisez ensuite `boussole.som_brut_actif: true`. `--som-rafraichir` (v1.17.0) est un alias sans effet, conservé pour la compatibilité rétroactive.

Depuis la version v1.17.2, l'injecteur efface les marqueurs laissés par une capture précédente `--som` sur la même page avant de renombrer — sans cela, un élément masqué ou défilé entre deux captures pourrait conserver un identifiant obsolète `data-dw-som-id`, ce qui pourrait entrer en conflit avec un élément nouvellement numéroté et entraîner une résolution incorrecte.

7k. Site bloqué par le WAF (erreur 403 immédiate)

```
# Essayez discrètement.
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url https://target.local/ --mode fast --stealth
```

Si l'erreur 403 persiste avec `--stealth`: le site utilise l'empreinte TLS (JA3/JA4) ou une analyse comportementale avancée (Cloudflare Enterprise). `playwright-stealth` ne contourne pas ces protections. Voir `docs/RETOUR_EXPERIENCE.md` FR-77/FR-78/FR-79 pour plus de contexte.

Diwall signale également un blocage probable de manière passive, sans que vous ayez à vérifier vous-même le code d'état HTTP — voir la section 3e (`respect.waf_bloquants`).

7l. La navigation initiale ne se termine jamais — `--wait-until` (v1.22.0)

Symptôme : `TimeoutError` lors de la navigation initiale, et l'augmentation de `--timeout` ne change rien (45 s échoue exactement comme 10 s). Cause : par défaut, Diwall attend `networkidle` – 500 ms de silence réseau. Une page qui interroge continuellement (statistiques en direct, compteurs actualisés automatique-

ment, panneaux d'administration du routeur) ne produit jamais ce silence, donc aucune valeur de délai d'attente ne peut être suffisamment grande.

```
# shot.py — direct reconnaissance
/opt/diwall/venv/bin/python3 /opt/diwall/shot.py \
  --url http://target.local/ --wait-until load --som --a11y --guide-version 1.3

# rpa.py — propagé à shot.py, de sorte que les scénarios atteignent les mêmes cibles.
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario ./admin_login.json --wait-until load --guide-version 1.3
```

Un scénario peut également l'inclure comme propriété racine, ce qui le rend autonome :

```
{"url": "http://target.local/", "wait_until": "load", "actions": [...]}
```

Le paramètre de ligne de commande a priorité sur la propriété du scénario.

Valeur	Attend	Utiliser lorsque
`networkidle`	500 ms de silence réseau	par défaut - à conserver sauf en cas de problème
`load`	événement `load` (page et ressources secondaires)	sondage continu / statistiques en direct
`domcontentloaded`	HTML analysé, les ressources secondaires sont toujours en attente	page très lourde, seul le DOM est nécessaire

S'applique uniquement à la navigation initiale ; l'action `naviguer` n'est pas affectée. `boussole.wait_until` affiche la valeur uniquement lorsqu'elle diffère de la valeur par défaut.

---

8. Surveillance visuelle — watch.py

8a. Sauvegarder une référence

```
/opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
  --url https://target.local/status \
  --sauver-reference \
  --nom home
```

La référence est enregistrée dans `/opt/diwall/references/`.

8b. Comparaison avec la référence (différence de pixels)

```
/opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
  --url https://target.local/status \
  --comparer-pixel /opt/diwall/references/target.local_home/reference.png \
  --nom home
```

Jugements :

`taux_diff`	Verdict	Code de sortie
< 0,2 %	`stable`	0
0,2 % – 5 %	`drift`	0
≥ 5 %	`regression`	1
Dimensions différentes	`viewport_mismatch`	2

8c. Comparaison sémantique (LLM)

```
/opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
```

```
  --url https://target.local/status \
  --comparer \
  --llm local
```

Combiner l'analyse des différences de pixels et l'analyse par modèle linguistique (LLM) :

```
--llm-en-complement   # LLM only if pixel verdict is drift or regression
```

`--llm claude` (v1.24.2) envoie les deux captures — la référence et l'actuelle — à l'API Anthropic au lieu du modèle local ; il s'applique à `--comparer` comme à `--llm-en-complement`. Les captures quittent la machine, réduites pour que leur plus grand côté ne dépasse pas 1568 px, la taille à laquelle l'API travaille. Il faut le module `anthropic`, absent d'une installation standard, et une clé dans l'environnement du processus :

```
sudo /opt/diwall/venv/bin/pip install anthropic
ANTHROPIC_API_KEY=... diwall-watch --url https://target.local/status --comparer --llm
 ↪  claude --guide-version 1.3
```

Un module absent, une clé refusée ou un refus du modèle revient sous forme de message dans la sortie (`erreur`, ou `analyse_llm` en mode pixel), jamais sous forme de trace Python.

8d. Ignorer une zone animée

```
/opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
  --url https://target.local/status \
  --comparer-pixel reference.png \
  --exclure-zone 100,200,300,50    # X,Y,Width,Height in pixels
```

8e. Boucle de surveillance

```
while true; do
  /opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
    --url https://target.local/status \
    --comparer-pixel /opt/diwall/references/status-ok.png \
    --ntfy-url https://ntfy.sh/my-alerts
  sleep 60
done
```

8f. Cron pour la surveillance autonome

```
# /etc/cron.d/diwall-monitor
*/30 * * * * diwall /opt/diwall/venv/bin/python3 /opt/diwall/watch.py \
  --url https://target.local/status \
  --comparer-pixel /opt/diwall/references/status-ok.png \
  --ntfy-url https://ntfy.sh/my-alerts \
  >> /var/log/diwall/cron.jsonl 2>&1
```

8g. Surveillance structurelle continue — `monitor-verifier.sh` (v1.18.0)

Compléments 8a–8f : `watch.py` surveille l'*apparence* (pixels/sémantique). `scripts/monitor-verifier.sh` surveille la *structure* (`http_status`, `dom_stats`, `evaluations`, nombre de SoM) — image nulle, appel LLM nul, basé sur `--no-capture` + `--replay-verifier` (section 5h).

```
# Premier lancement : créer la référence structurelle.
/opt/diwall/venv/bin/python3 /opt/diwall/rpa.py \
  --scenario /opt/diwall/scenarios/sillage_login.json \
  --sauver-verifier-reference /opt/diwall/references/sillage_login.ref.json

# Un script de vérification et d'alerte — ce n'est pas un démon, exécutez-le à plusieurs
 ↪  reprises via cron.
# Les fichiers `*.sh` situés dans le répertoire `scripts/` ne sont jamais déployés vers
 ↪  /opt/diwall/, ils s'exécutent donc depuis Git.
```

```
# source, en tant que votre propre utilisateur.
bash ~/git/Diwall/Diwall/scripts/monitor-verifier.sh \
  --scenario /opt/diwall/scenarios/sillage_login.json \
  --reference /opt/diwall/references/sillage_login.ref.json \
  --ntfy-topic diwall-monitoring

# crontab -e (votre propre fichier crontab)
*/15 * * * * bash ~/git/Diwall/Diwall/scripts/monitor-verifier.sh \
  --scenario /opt/diwall/scenarios/sillage_login.json \
  --reference /opt/diwall/references/sillage_login.ref.json \
  --ntfy-topic diwall-monitoring \
  >> /var/log/diwall/cron-structural.jsonl 2>&1
```

Stable → silence. Régression → une notification `ntfy` avec le différentiel. Chaque invocation est un processus isolé : pas de démon, pas de risque de fuite mémoire, et les plafonds de Navigation Respectueuse se réinitialisent proprement à chaque passe.

---

9. Journal des opérations

Le journal est configurable dans `diwall.conf` (v1.15.0) :

```
"journal": {
  "chemin": "~/Vaults/__PROJET__/Diwall/operations.jsonl"
}
```

Si le fichier est absent ou si le répertoire chiffré n'est pas monté, utiliser comme solution de repli : la variable d'environnement `DIWALL_JOURNAL`, puis `/var/log/diwall/operations.jsonl`.

```
# Lisez les 10 dernières entrées.
tail -n 10 ~/Vaults/__PROJET__/Diwall/operations.jsonl | python3 -m json.tool

# Filtrez par cible (journal.py outil).
/opt/diwall/venv/bin/python3 /opt/diwall/journal.py \
  --cible app.example.com

# Filtrez uniquement les opérations qui modifient l'état.
/opt/diwall/venv/bin/python3 /opt/diwall/journal.py \
  --cible app.example.com --mutatif

# À partir d'une date.
/opt/diwall/venv/bin/python3 /opt/diwall/journal.py \
  --cible app.example.com --depuis 2026-07-01

# Exécutions ayant échoué uniquement (v1.20.0) — resultat != "succès"
/opt/diwall/venv/bin/python3 /opt/diwall/journal.py \
  --cible app.example.com --erreurs
```

Champs dans chaque entrée :

Champ	Signification
`ts`	Horodatage ISO 8601
`version`	Version Diwall
`outil`	`shot.py` ou `rpa.py`
`cible_url`	URL cible
`scenario`	Chemin du fichier de scénario (mode RPA)
`source_scenario`	Nom du fichier de scénario uniquement, sans chemin (v1.18.0) — active `mode_conseille` (section 2e)
`resultat`	`"succes"` ou `"echec"`
`mutatif`	`true` si au moins une action d'écriture est présente

Champ	Signification
`duree_ms`	Durée en ms
`intention`	Étiquette transmise via `--intention` ou champ de scénario `intention`

9a. Rotation des journaux (G-36, CHANTIER_SANITISATION.md)

Diwall ne fournit pas de configuration `logrotate` — `/var/log/diwall/operations.jsonl` et grandit indéfiniment jusqu'à ce que l'administrateur en installe une. `lib/journal.py` ouvre et ferme le fichier à chaque écriture (pas de descripteur de fichier persistant entre les exécutions), spécifiquement pour que le comportement par défaut de `logrotate` (renommer le fichier actuel, créer un nouveau fichier) fonctionne correctement sans aucune option spéciale : la prochaine écriture rouvre le chemin et trouve le nouvel inode.

Ne pas ajouter `copytruncate` à une configuration de logrotate pour Diwall, car c'est inutile ici (contrairement aux outils qui maintiennent un descripteur de fichier ouvert pendant toute leur durée de vie) et cela réintroduit une fenêtre de perte d'écriture que cette conception a été conçue pour éviter. Exemple `/etc/logrotate.d/diwall`:

```
/var/log/diwall/operations.jsonl {
    weekly
    rotate 8
    compress
    delaycompress
    missingok
    notifempty
    create 0640 diwall diwall
}
```

`journal.py` (le lecteur) suit déjà les fichiers rotatifs de manière transparente. (`operations.jsonl`, `.1`, `.2.gz`, ...) — aucune étape supplémentaire n'est nécessaire après la rotation.

---

10. Options de ligne de commande — référence

shot.py

Flag	Default	Description
`--version`	—	Affiche la version installée et se termine immédiatement — aucun Playwright, aucun autre argument requis (v1.18.0)
`--guide-version X.Y`	—	Preuve de lecture de `docs/GUIDE_LLM.md` — requis sauf si un marqueur local valide existe déjà (v1.18.0, section 1)
`--url URL`	required	URL à capturer
`--actions FILE`	—	Fichier JSON des actions séquentielles
`--output-dir DIR`	`/tmp/diwall`	Répertoire de sortie PNG
`--timeout MS`	10000	Délai d'attente Playwright par action (ms)
`--screenshot-timeout MS`	120000	Délai d'attente pour `page.screenshot()` (ms). Différent de `--timeout`
`--largeur PX`	1280	Largeur de la fenêtre d'affichage
`--hauteur PX`	720	Hauteur de la fenêtre d'affichage

Flag	Default	Description
`--som`	off	Active le marquage (numérotation des éléments)
`--a11y`	off	Inclut l'arborescence d'accessibilité dans le fichier JSON
`--shadow-dom`	off	Parcourt les Shadow Roots pour le marquage (Angular, Lit, Stencil)
`--stealth`	off	Mode furtif playwright-stealth (v1.15.0)
`--mode fast|full`	—	`fast` = `--no-capture --a11y`. `full` = comportement par défaut
`--no-capture`	off	Ignore la capture PNG et le marquage
`--llm local|claude`	`local`	Moteur LLM pour `cliquer_visuel`
`--secrets FILE`	—	Chemin explicite vers un fichier de crédentielles
`--auth-indicator SEL`	—	Sélecteur CSS présent uniquement dans la session authentifiée
`--auth-indicator-negative SEL`	—	Sélecteur CSS présent uniquement en dehors de la session authentifiée
`--intention TEXT`	—	Étiquette commerciale enregistrée dans le journal
`--sauver-session FILE`	—	Enregistre les cookies après les actions
`--reprendre-session FILE`	—	Reprend une session enregistrée
`--interval-capture N`	0	Captures périodiques toutes les N secondes pendant `attendre`, `pause`
`--som-brut`	off	Force une réindexation pure du marquage ; désactive le chemin stable et la détection de divergence du résolveur hybride (v1.24.0, section 7j)
`--som-rafraichir`	off	Alias sans effet depuis v1.24.0 — la résolution hybride est le comportement par défaut (v1.17.0, section 7j)
`--ignorer-waf`	off	Un bloc WAF détecté dégrade `niveau_confiance` mais ne force plus `pret_a_agir: false` par lui-même (v1.17.2, section 3e)
`--http-credentials`	off	Résout les informations d'identification HTTP Basic Auth à partir du fichier de crédentielles, limitées à l'origine de la cible (v1.21.0, section 4g)
`--no-evaluer`	off	Refuse l'action **evaluer** pour toute l'exécution — recommandé en production pour les cibles avec des formulaires sensibles (v1.15.1)

Flag	Default	Description
`--no-filtre-evaluer`	off	Désactive la neutralisation de la sortie standard des valeurs de retour de **evaluer**, des URL et des messages d'erreur — uniquement pour les exécutions de débogage explicites. La neutralisation est activée par défaut ; lorsqu'elle est désactivée, `boussole.filtre_evaluer_actif: false` est défini dans la sortie afin que l'opérateur puisse l'auditer directement à partir du fichier JSON (v1.23.0)

rpa.py

Transmet tous les drapeaux shot.py pertinents, ainsi que :

Flag	Description
`--version`	Affiche la version installée et se termine immédiatement (v1.18.0)
`--guide-version X.Y`	Preuve de lecture de `docs/GUIDE_LLM.md` — vérifiée indépendamment, même règle que shot.py (v1.18.0)
`--scenario FILE`	Chemin vers le scénario JSON ou YAML (obligatoire)
`--url URL`	Remplace l'URL du scénario sans modifier le fichier
`--stealth`	Propagé à shot.py
`--mode fast|full`	Propagé à shot.py
`--som-brut`	Propagé à shot.py. Peut également être défini comme propriété racine du scénario `"som_brut": true` (v1.24.0, section 7j)
`--som-rafraichir`	Propagé à shot.py — alias sans effet depuis v1.24.0 (v1.17.0, section 7j)
`--ignorer-waf`	Propagé à shot.py (v1.17.2, section 3e)
`--http-credentials`	Propagé à shot.py. Peut également être défini comme propriété racine du scénario `"http_credentials": true` (v1.21.0, section 4g)
`--sauver-verifier-reference FILE`	Enregistre la référence structurelle pour `--replay-verifier` (v1.17.0, section 5h)
`--replay-verifier FILE`	Compare l'exécution à une référence structurelle, sortie 1 en cas de régression (v1.17.0, section 5h)
`--checkpoint FILE`	Reprend un scénario long après un échec pendant l'exécution (v1.17.0, section 5i)

watch.py

Flag	Description
`--version`	Affiche la version installée et se termine immédiatement (v1.18.0)
`--guide-version X.Y`	Preuve de lecture de `docs/GUIDE_LLM.md` — vérifiée indépendamment, même règle que shot.py (v1.18.0)
`--url URL`	URL à surveiller
`--sauver-reference`	Capture et sauvegarde comme référence
`--comparer-pixel REF`	Différence de pixels par rapport au fichier PNG REF
`--comparer`	Différence sémantique LLM

Flag	Description
`--llm local|claude`	Moteur pour `--comparer` et `--llm-en-complement` (par défaut `local`; `claude` envoie les captures à l'API Anthropic, v1.24.2)
`--nom NAME`	Nom de la vue (plusieurs vues par URL)
`--seuil-stable F`	Seuil `stable` (par défaut : 0.002 = 0.2%)
`--seuil-regression F`	Seuil `regression` (par défaut : 0.05 = 5%)
`--exclure-zone X,Y,W,H`	Zone à ignorer (répétable)
`--heatmap`	Produit une image PNG des zones modifiées
`--ntfy-url URL`	Envoie une alerte ntfy en cas de régression
`--llm-en-complement`	Ajoute une différence LLM lorsque le pixel = dérive ou régression

---

11. Codes de sortie et résultats

Codes de sortie

Code	Cause	Que faire
0	Succès	—
1	Erreur Playwright, action en échec, assertion rpa.py	Lire `erreur` dans le JSON. Voir `GUIDE_LLM_INTERACTIONS.md`
1	`guide_non_lu` — `--guide-version` absent ou erroné, aucun marqueur valide (v1.18.0)	Se déclenche avant le lancement de Playwright. Lire `docs/GUIDE_LLM.md`, relancer avec `--guide-version X.Y` (section 1)
2	`viewport_mismatch` (watch.py)	Reprendre la référence au même viewport
3	Module `playwright` introuvable	Invoquer via `/opt/diwall/venv/bin/python3`
42	`SecretsFermesError` — répertoire chiffré non monté, ou somme de contrôle invalide	Le monter, ou vérifier le fichier d'identifiants
43	`SecretsNonConfigureError` — `diwall.conf` absent	`sudo cp /opt/diwall/diwall-sample.conf /opt/diwall/diwall.conf && sudo nano /opt/diwall/diwall.conf`

Structure du JSON de sortie

```
{
  "succes": true,
  "http_status": 200,
  "url_finale": "https://target.local/dashboard",
  "erreurs_js": [],
  "erreurs_console": [],
  "duree_ms": 2400,
  "horodatage": "2026-07-01T12:00:00+02:00",
  "capture": "/tmp/diwall/a1b2c3d4e5f6/capture_1234567890123456789.png",
  "capture_som": "/tmp/diwall/a1b2c3d4e5f6/capture_som_1234567890123456789.png",
  "elements_som": [...],
  "a11y_tree": "...",
  "evaluations": [...],
  "latences_actions": [
```

```
    {"index": 0, "type": "naviguer", "latence_ms": 842},
    {"index": 1, "type": "cliquer_som", "latence_ms": 63}
  ],
  "respect": {
    "pages_visitees": 0,
    "actions_executees": 3,
    "duree_totale_ms": 2400,
    "indice_agressivite": 0.33
  },
  "etat": {
    "pret_a_agir": true,
    "niveau_confiance": "eleve",
    "raisons": ["aucun signal de friction détecté"]
  },
  "boussole": {
    "utilisateur": "operator",
    "ip_locale": "__IP_LAN__",
    "repertoire": "/opt/diwall",
    "operation_id": "a1b2c3d4e5f6",
    "url_courante": "https://target.local/dashboard",
    "titre_page": "Dashboard — My App",
    "stealth_actif": true,
    "shadow_dom_actif": true,
    "auth_status": "active",
    "som_hors_viewport": 0,
    "respect": { "pages_visitees": 0, "actions_executees": 3, "duree_totale_ms": 2400,
    ↪  "indice_agressivite": 0.33, "som_resolution": "stable" }
  },
  "diwall_meta": {
    "version_shot": "1.23.0",
    "profil": "operator",
    "modeles_appeles": []
  }
}
```


`operation_id` (v1.16.0) est toujours présent et identifie de manière unique cette exécution ; il nomme le répertoire d'isolation sous `/tmp/diwall/<operation_id>/` et correspond au champ `operation_id` de l'entrée de cette exécution dans le journal des opérations (section 9). `etat` (v1.16.0) est présent uniquement sur le chemin de succès. `latences_actions` (v1.20.0) est toujours présent (liste vide s'il n'y a pas d'actions), une entrée par action réellement exécutée ; voir `GUIDE_LLM_MONITORING.md` pour comprendre comment il complète `respect.duree_totale_ms`.

Les clés conditionnelles (absentes lorsqu'elles sont inactives) : `capture`, `capture_som`, `elements_som`, `a11y_tree`, `evaluations`, `auth_status`, `stealth_actif`, `shadow_dom_actif`, `som_brut_actif`, `som_hors_viewport`, `session_derive`, `respect.plafond_atteint`, `respect.waf_bloquants`, `respect.som_resolution` (présentes chaque fois qu'une action SoM a été résolue), `respect.som_derive_detectee` (présentes uniquement lors d'une divergence stable/brute), `respect.indice_agressivite` (présentes chaque fois qu'au moins une action a été exécutée), `actions_executees_avant_echec`, `pages_visitees_avant_echec` (uniquement dans le fichier JSON d'erreur, v1.17.0), `etat.mode_conseille` (présentes uniquement avec des données de priorité réelles `diagnostic_dom.json` pour cet hôte, v1.18.0, section 2e).

Erreur — format


```
{
  "succes": false,
  "erreur": "secrets_fermes",
  "message": "Le répertoire chiffré Diwall est initialisé mais non monté.",
  "code_sortie_recommande": 42,
```

```
  "boussole": { "url_courante": "", "titre_page": "" }
}
```

Chemins de référence

Chemin	Rôle
`/opt/diwall/`	Installation de production
`/opt/diwall/venv/bin/python3`	Python à utiliser pour chaque invocation
`/opt/diwall/diwall.conf`	Configuration de la machine (identifiants, navigation, logs)
`/opt/diwall/diwall-sample.conf`	Modèle de configuration
`/opt/diwall/scenarios/`	Scénarios RPA
`/opt/diwall/docs/`	Documentation
`/opt/diwall/references/`	Références visuelles watch.py
`/tmp/diwall/<operation_id>/`	Captures temporaires pour une seule exécution, isolées par `operation_id` (v1.16.0, effacées au redémarrage)
`~/Vaults/__PROJET__/Diwall/`	Identifiants + logs (volume gocryptfs)
`~/git/Diwall/Diwall/`	Sources Git (modifier ici, puis `deploy.sh`)

Déployez après avoir modifié les sources :

```
bash ~/git/Diwall/Diwall/scripts/deploy.sh
```