MANUEL TECHNIQUE COMPLET

SYNOVEX

Sonde de supervision applicative autonome pour Raspberry Pi 5

Matériel : Raspberry Pi 5

Éditions : Essential, Professional, Enterprise



SYNOVEX — SIREN 884 713 009

Table des matières

Informations sur le document 8

PARTIE I — PRÉSENTATION DE LA SOLUTION 11

Chapitre 1 — Introduction 12
- 1.1 Présentation générale du produit 12
- 1.2 Objectifs du manuel 13
- 1.3 Public concerné 14
- 1.4 Périmètre de la solution 14
- 1.5 Organisation du manuel 16
- 1.6 Statut du document 16
- 1.7 Conventions de lecture 17
- 1.8 Parcours de lecture conseillés 19
- 1.9 Support et retour d'expérience 20

Chapitre 2 — Comprendre la supervision applicative 21
- 2.1 Qu'est-ce que la supervision ? 21
- 2.2 Disponibilité, performance et fiabilité 21
- 2.3 Les niveaux de supervision 23
- 2.4 Pourquoi un serveur disponible ne garantit pas le bon fonctionnement d'une application 24
- 2.5 Limites des contrôles techniques classiques 26
- 2.6 L'intérêt de la supervision synthétique 27
- 2.7 Détection proactive et réduction du temps de détection 29
- 2.8 Une supervision complémentaire, pas un remplacement 31
- 2.9 L'apport de la supervision applicative aux équipes 32

Chapitre 3 — Architecture de la solution 33
- 3.1 Principes d'architecture 33
- 3.2 Vue d'ensemble 33
- 3.3 Composants de la sonde 36
- 3.4 Stockage et conservation des données 39
- 3.5 Cycle de vie d'une exécution 39
- 3.6 Publication et interopérabilité des résultats 41
- 3.7 Administration de la sonde 42
- 3.8 Modes de fonctionnement 43
- 3.9 Sécurité intégrée 44
- 3.10 Comparaison avec l'architecture historique 45

PARTIE II — INSTALLATION ET MISE EN SERVICE 46

Chapitre 4 — Prérequis 47
- 4.1 Matériel compatible 47
- 4.2 Alimentation électrique 49
- 4.3 Carte microSD 50
- 4.4 Refroidissement et boîtier 52
- 4.5 Réseau 52
- 4.6 Prérequis de sécurité et de gouvernance 55
- 4.7 Poste d'administration 56
- 4.8 Conditions d'installation physique 56
- 4.9 Liste de vérification avant démarrage 57

Chapitre 5 — Installation physique 59
5.1 Vue d'ensemble de la procédure 60
5.2 Précautions de manipulation 61
5.3 Identification du matériel 62
5.4 Assemblage avec la microSD préconfigurée 64
5.5 Mise en place du boîtier complet 69
5.6 Choix de l'emplacement 72
5.7 Connexion Ethernet 73
5.8 Branchement de l'alimentation et démarrage 74
5.9 Vérifications initiales 74
5.10 Erreurs fréquentes lors de l'installation 76

Chapitre 6 — Premier démarrage et mise en service 78
6.1 Les deux méthodes de mise en service 78
6.2 Déroulement de l'initialisation 81
6.3 Mise en service par fichier de provisionnement (méthode A) 82
6.4 Mise en service par l'assistant web (méthode B) 91
6.5 Édition de la sonde 97
6.6 Vérifier la disponibilité de la sonde 97
6.7 Premier contrôle de supervision 99
6.8 Vérifier la remontée des résultats 101
6.9 Checklist de fin de mise en service 102

Chapitre 7 — Configuration réseau et sécurité 104
7.1 Principes de sécurité de la sonde 104
7.2 Adressage IP 104
7.3 Résolution DNS 115
7.4 Synchronisation de l'heure 115
7.5 Proxy sortant 116
7.6 Flux réseau et pare-feu 118
7.7 Certificats et protocoles sécurisés 120
7.8 Identifiants, secrets et comptes techniques 122
7.9 Sécurisation des accès administrateur 123
7.10 Segmentation et positionnement réseau 126
7.11 Sécurité physique 128
7.12 Journalisation et audit 128
7.13 Recommandations aux RSSI 129

PARTIE III — UTILISATION DE LA SONDE 131

Chapitre 8 — Principes de fonctionnement 132
8.1 La notion de contrôle 132
8.2 Les cibles supervisées 135
8.3 La fréquence d'exécution 136
8.4 Le délai maximal d'exécution (timeout) 137
8.5 Nouvelles tentatives et confirmation d'un incident 138
8.6 Les seuils de performance 141
8.7 Le résultat d'une exécution 141
8.8 Les états et les erreurs 143
8.9 Mesures de performance 147
8.10 Erreurs d'exécution et échecs de la cible 147
8.11 Classes d'exécution et concurrence 148
8.12 Rétention et exploitation des données 149
8.13 Incident réseau, applicatif ou fonctionnel 150
8.14 Fonctionnement autonome 151

Chapitre 9 — Administration de la sonde 154
9.1 Moyens d'administration 154
9.2 Interface d'administration 156
9.3 API REST : principes 157
9.4 Utilisateurs et rôles 163
9.5 Jetons d'API 166
9.6 Gestion des contrôles 168
9.7 Scripts de scénarios 171
9.8 Secrets 172
9.9 Paramètres de la sonde 175
9.10 Journal d'audit 178
9.11 Événements en temps réel 179
9.12 Sauvegarde et restauration de la configuration 179
9.13 Automatisation 181
9.14 Gestion centralisée de plusieurs sondes 185

Chapitre 10 — Contrôles réseau et techniques 186
10.1 Vue d'ensemble 186
10.2 Choisir le bon contrôle 188
10.3 Contrôle ICMP (icmp) 190
10.4 Contrôle TCP (tcp) 193
10.5 Contrôle DNS (dns) 196
10.6 Contrôle HTTP et HTTPS (http) 199
10.7 Plugins Nagios (nagios) 208
10.8 Synthèse 214

Chapitre 11 — Contrôles applicatifs et métier 215
11.1 De la disponibilité technique à la réussite fonctionnelle 215
11.2 Superviser une API REST avec le contrôle http 216
11.3 Enchaîner des requêtes avec le contrôle api 218
11.4 Superviser un web service SOAP 222
11.5 Concevoir des contrôles fonctionnels robustes 224
11.6 Contrôles de bases de données (database) 224
11.7 Indicateurs métier 233
11.8 Détection d'anomalies 234
11.9 Précautions et sécurité 236
11.10 Synthèse 237

Chapitre 12 — Planification et plans d'exécution 238
12.1 Deux façons d'organiser les exécutions 238
12.2 Planifier un contrôle 239
12.3 Les plans d'exécution 243
12.4 Comportement d'un plan 244
12.5 Créer et piloter un plan 247
12.6 Charge et dimensionnement 249
12.7 Exemples de configurations 251
12.8 Interventions planifiées et fenêtres de maintenance 254

PARTIE IV — SCÉNARIOS DE NAVIGATION AVEC PUPPETEER 255

Chapitre 13 — Introduction à Puppeteer 256
13.1 Présentation de Puppeteer 256
13.2 Chromium et le protocole Chrome DevTools 256
13.3 Le navigateur headless 257
13.4 Navigation automatisée et simple requête HTTP 258

13.5 Cas d'utilisation en supervision 260
13.6 Avantages et limites 260
13.7 De CasperJS à Puppeteer 261
13.8 Prérequis techniques dans la sonde 263

Chapitre 14 — Construction d'un scénario 266
14.1 Deux modes d'écriture 266
14.2 Paramètres d'un contrôle navigateur 267
14.3 Scénario déclaratif 268
14.4 Concevoir un scénario avec l'enregistreur de Chrome 271
14.5 Scénario scripté : structure et contrat 272
14.6 Ouvrir une page et naviguer 275
14.7 Rechercher des éléments 276
14.8 Interagir avec la page 278
14.9 Authentification 279
14.10 Gérer l'attente et les délais 280
14.11 Assertions 282
14.12 Traitement des erreurs 283
14.13 Iframes, fenêtres, boîtes de dialogue et téléchargements 285
14.14 Fermeture et libération des ressources 286
14.15 Tester sur poste de travail avec le lanceur local 287
14.16 Déployer un scénario sur la sonde 290
14.17 Cycle de vie et versionnage des scripts 293

Chapitre 15 — Exemples progressifs 294
15.1 Présentation des exemples 294
15.2 Exemple 1 — Vérifier qu'une page web est accessible 297
15.3 Exemple 2 — Vérifier la présence d'un élément attendu 299
15.4 Exemple 3 — S'authentifier sur une application 301
15.5 Exemple 4 — Parcourir plusieurs pages 305
15.6 Exemple 5 — Vérifier le fonctionnement d'un moteur de recherche 308
15.7 Exemple 6 — Ajouter un produit au panier 310
15.8 Exemple 7 — Vérifier un parcours de commande en environnement de test 313
15.9 Exemple 8 — Détecter une erreur fonctionnelle 316
15.10 Synthèse des essais 321
15.11 Adapter les exemples à vos applications 321

Chapitre 16 — Bonnes pratiques des scénarios 323
16.1 Utiliser des sélecteurs robustes 323
16.2 Écrire des assertions explicites 324
16.3 Maîtriser les délais 324
16.4 Attendre des événements plutôt que temporiser 325
16.5 Gérer les erreurs sans les masquer 325
16.6 Limiter la consommation de mémoire et de processeur 326
16.7 Gérer les sessions et les comptes de test 326
16.8 Isoler les scénarios 327
16.9 Nettoyer les ressources et les données 327
16.10 Protéger les secrets 328
16.11 Maîtriser les ressources tierces 328
16.12 Identifier le trafic de la sonde 329
16.13 Captures d'écran, traces et données personnelles 329
16.14 Fiabilité dans la durée 330
16.15 Maintenir les scénarios au rythme des évolutions applicatives 331
16.16 Collaborer avec les équipes de développement 332
16.17 Liste de contrôle avant la mise en service d'un scénario 332

PARTIE V — EXPLOITATION ET MAINTENANCE 334

Chapitre 17 — Consultation et interprétation des résultats 335

17.1 Où se trouvent les résultats 335
17.2 Consultation dans l'interface d'administration 336
17.3 Consultation par l'API REST 338
17.4 Lire un résultat d'exécution 342
17.5 Méthode d'analyse d'un échec 344
17.6 Bagotage, nouvelles tentatives et confirmation des incidents 348
17.7 Indicateurs : temps de réponse, percentiles et disponibilité 349
17.8 Historique, rétention et agrégats 352
17.9 État de santé de la sonde 352
17.10 Exporter les données 355
17.11 Synthèse 356

Chapitre 18 — Intégration aux plateformes d'observabilité 358

18.1 Une approche fondée sur les standards ouverts 358
18.2 Ingestion n'est pas intégration 361
18.3 Conventions de télémétrie 361
18.4 Configurer une destination d'export 368
18.5 Prometheus 371
18.6 Grafana 378
18.7 Datadog 380
18.8 Dynatrace 382
18.9 OpenTelemetry Collector du client 384
18.10 Nagios et Centreon 386
18.11 Journaux d'audit vers un SIEM 391
18.12 Comportement en cas d'indisponibilité d'une destination 391
18.13 Sécurité des intégrations 393
18.14 Choisir une stratégie d'intégration 394
18.15 Exploitation des alertes 395
18.16 Dépannage des exports 410

Chapitre 19 — Diagnostic et résolution des problèmes 412

19.1 Méthode générale de diagnostic 412
19.2 Outils de diagnostic disponibles 414
19.3 Démarrage et matériel 418
19.4 Mise en service 421
19.5 Réseau 423
19.6 Contrôles techniques et applicatifs 425
19.7 Scénarios navigateur (Puppeteer) 428
19.8 Destinations d'export et intégrations 430
19.9 Stockage 432
19.10 Mises à jour 433
19.11 Types d'erreur et actions de premier niveau 435
19.12 Préparer une demande au support 436

Chapitre 20 — Maintenance et mises à jour 439

20.1 Répartition des responsabilités 439
20.2 Arrêt et redémarrage 439
20.3 Protection du support de stockage 441
20.4 Mises à jour 444
20.5 Sauvegarde et restauration de la configuration 470
20.6 Remplacer une carte microSD ou une sonde 472
20.7 Surveillance de la santé de la sonde 475
20.8 Rotation des secrets et des jetons 476

20.9 Maintenance des scénarios et des contrôles 477
20.10 Réinitialisation aux paramètres d'usine 478
20.11 Plan de maintenance préventive 481
20.12 Entretien physique 481
20.13 Fin de vie et effacement sécurisé 482

ANNEXES — RÉFÉRENCES, GLOSSAIRE ET INFORMATIONS PRATIQUES 483

Chapitre 21 — Annexes 484
21.1 Annexe A — Glossaire 484
21.2 Annexe B — Questions fréquentes 493
21.3 Annexe C — Récapitulatif des prérequis 497
21.4 Annexe D — Checklist de mise en service 498
21.5 Annexe E — Tableau des flux réseau 500
21.6 Annexe F — Référence de l'API REST 502
21.7 Annexe G — Référence des configurations 507
21.8 Annexe H — Types d'erreur 512
21.9 Annexe I — États et codes retour Nagios 514
21.10 Annexe J — Correspondance avec la génération historique 514
21.11 Annexe K — Informations de version et statut des informations 516
21.12 Annexe L — Assistance et contacts 517
21.13 Annexe M — Licences tierces et mentions 518

SYNOVEX

Manuel technique complet

Sonde de supervision applicative autonome pour Raspberry Pi 5

Information	Valeur
Document	Manuel technique complet — identifiant SYN-DOC-00
Révision documentaire	B
Produit	SYNOVEX
Matériel pris en charge	Raspberry Pi 5
Éditions couvertes	SYNOVEX Essential, SYNOVEX Professional, SYNOVEX Enterprise
Éditeur	SYNOVEX — SIREN 884 713 009
Documentation en ligne	`https://docs-synovex.incitypark.fr` (après l'annonce de sa mise en service)
Langue	Français

Mentions légales

Éditeur

Cette documentation est éditée par SYNOVEX, SIREN 884 713 009. SYNOVEX est aussi le nom commercial de la gamme de sondes de supervision décrite.

Propriété intellectuelle et diffusion

© SYNOVEX. Tous droits réservés. Les textes, schémas, illustrations et exemples de cette documentation sont protégés par le droit d'auteur.

Cette documentation est réservée aux clients titulaires d'un droit d'utilisation du produit SYNOVEX. Le client peut la reproduire et la diffuser à l'intérieur de son organisation, et la communiquer aux prestataires qui interviennent pour son compte, pour les seuls besoins de l'installation, de l'exploitation et de la maintenance de ses sondes. Toute autre reproduction, diffusion ou mise à disposition, totale ou partielle, requiert l'accord écrit préalable de SYNOVEX.

Les fichiers d'exemple fournis avec la documentation (scénarios, règles d'alerte, tableau de bord, schémas JSON) peuvent être copiés et adaptés par le client pour l'exploitation de ses sondes.

Limitation de responsabilité

Les informations de cette documentation peuvent être modifiées sans préavis. L'éditeur s'efforce d'en assurer l'exactitude, mais ne peut être tenu responsable des erreurs ou omissions, ni des conséquences de l'utilisation des exemples fournis. Les exemples de configuration et de scénarios sont **fictifs** et doivent être adaptés à votre environnement, puis validés, avant tout usage. Cette documentation ne constitue pas un engagement contractuel : les engagements de l'éditeur, notamment de garantie, de support et de maintenance, résultent exclusivement de votre contrat.

Marques

SYNOVEX est un nom commercial de l'éditeur. Raspberry Pi est une marque de Raspberry Pi Ltd. Orange Pi est une marque de Shenzhen Xunlong Software Co., Ltd. Google Chrome est une marque de Google LLC ; Chromium et Puppeteer sont des projets open source de Google LLC. Java, OpenJDK, Oracle et MySQL sont des marques d'Oracle et/ou de ses filiales. Node.js est une marque de l'OpenJS Foundation. Prometheus et OpenTelemetry sont des marques de The Linux Foundation. Grafana est une marque de Grafana Labs. Datadog est une marque de Datadog, Inc. Dynatrace est une marque de Dynatrace LLC. Nagios est une marque de Nagios Enterprises, LLC. Centreon est une marque de Centreon. Microsoft et SQL Server sont des marques du groupe Microsoft. PostgreSQL est une marque de PostgreSQL Community Association of Canada. MariaDB est une marque de MariaDB plc. Les autres noms de produits et de sociétés cités peuvent être des marques de leurs propriétaires respectifs. Leur mention dans cette documentation n'implique aucun partenariat ni aucune approbation.

Logiciels tiers

La sonde intègre des logiciels libres et des composants tiers, distribués selon leurs propres licences. Les principaux composants et leurs licences sont listés à l'annexe M ; l'inventaire complet des composants accompagne chaque version du logiciel.

Matériel et fin de vie

Le Raspberry Pi 5 et ses accessoires officiels sont fabriqués par Raspberry Pi Ltd, qui en publie les déclarations de conformité. Avec l'offre microSD préconfigurée, le matériel est choisi et assemblé par vos soins. Avec l'offre boîtier complet, les informations de conformité applicables figurent dans les documents joints au produit. En fin de vie, la sonde relève de la collecte des déchets d'équipements électriques et électroniques : ne la jetez pas avec les déchets ordinaires et confiez-la à une filière de collecte agréée, après avoir effacé ses données (§ 20.13).

À propos de cette édition

Ce manuel est la **référence documentaire intégrale** de SYNOVEX. Il fait partie de la bibliothèque documentaire SYNOVEX, dont les six guides spécialisés (démarrage rapide, installation et administration, utilisation et supervision, développement des scénarios, intégration et observabilité, référence technique et API) sont produits à partir des mêmes sources : un même passage est rigoureusement identique dans ce manuel et dans les guides qui le reprennent.

Cette édition décrit le fonctionnement de référence de **SYNOVEX** sur **Raspberry Pi 5**. Les valeurs propres à chaque version de l'image système (versions des composants embarqués, valeurs mesurées, précisions de mise en œuvre) figurent dans les notes de version qui l'accompagnent et, sur la sonde, dans les informations système exposées par l'API (`GET /api/v1/system`). Les exemples de configuration et de scénarios sont fictifs. Les fonctions prévues dans une version ultérieure sont signalées par un encadré **Évolution prévue**. Les conventions du document sont détaillées au § 1.6 et à l'annexe K.

Comment utiliser ce manuel

Vous souhaitez…	Consultez
Comprendre ce que fait la sonde et pourquoi	Partie I, chapitres 1 à 3
Préparer l'installation	Chapitre 4, puis la checklist de l'annexe D
Installer et mettre en service la sonde	Chapitres 5 et 6
Faire valider l'installation par votre équipe sécurité	Chapitre 7 et annexe E
Configurer vos premiers contrôles	Chapitres 8, 10 et 11
Administrer la sonde (comptes, jetons, secrets, API)	Chapitre 9
Organiser l'exécution des contrôles	Chapitre 12
Créer des parcours utilisateurs avec Puppeteer	Partie IV, chapitres 13 à 16
Lire et interpréter les résultats	Chapitre 17
Envoyer les résultats vers Prometheus, Grafana, Datadog, Dynatrace, Nagios ou Centreon	Chapitre 18
Résoudre un problème	Chapitre 19
Assurer la maintenance	Chapitre 20
Trouver une définition, une référence ou un contact	Chapitre 21 (annexes)

PARTIE I

Présentation de la solution

Chapitre 1 — Introduction

Chapitre 2 — Comprendre la supervision applicative

Chapitre 3 — Architecture de la solution

Chapitre 1 — Introduction

1.1 Présentation générale du produit

SYNOVEX est la gamme de sondes autonomes de supervision réseau, applicative et métier de l'éditeur. Elle est proposée en trois éditions, **SYNOVEX Essential**, **SYNOVEX Professional** et **SYNOVEX Enterprise** (§ 1.4.1), sous forme de carte microSD préconfigurée ou de boîtier prêt à l'emploi. Dans la suite de ce manuel, le produit est désigné simplement par « la sonde ».

La sonde est un équipement de supervision applicative conçu pour surveiller la disponibilité, les performances et le bon fonctionnement des services informatiques **depuis différents points d'observation**. Elle repose sur un ordinateur monocarte Raspberry Pi 5 et sur une carte microSD fournie préconfigurée.

Elle peut être déployée aussi bien au sein du réseau de votre organisation qu'à l'extérieur : sur un site distant ou dans une agence, chez un client, chez un partenaire ou un sous-traitant, ou dans différentes régions géographiques. Plus généralement, elle s'installe dans tout environnement qui dispose des accès réseau nécessaires aux contrôles configurés. Cette flexibilité permet de superviser vos services depuis différents environnements réseau et zones géographiques, afin de vérifier leur accessibilité et leurs performances telles qu'elles sont réellement perçues par leurs utilisateurs.

Une fois mise en service, la sonde exécute automatiquement, à intervalles réguliers, les contrôles configurés. Elle enregistre les résultats et les met à disposition de vos équipes et de vos outils de supervision.

Cette approche est celle de la **supervision synthétique distribuée** (chapitre 2, § 2.6.1) : il ne s'agit pas seulement de vérifier qu'un service fonctionne depuis le réseau de l'entreprise, mais de s'assurer qu'il est réellement disponible et performant depuis les points d'accès de ses clients, de ses utilisateurs et de ses partenaires. Une sonde installée dans les locaux d'un client au Viêt Nam indique ainsi si vos applications sont accessibles et rapides **depuis ce réseau**, à travers ses serveurs DNS, son proxy, son opérateur et ses liens internationaux, ce qu'une mesure prise depuis votre propre infrastructure ne permet pas de connaître. Plusieurs sondes installées à des emplacements différents vous permettent de comparer ces points de vue ; les écarts entre leurs résultats sont des indices de localisation, que vos équipes confirment par l'analyse (chapitre 3, § 3.2.1).

Elle répond à une question que les outils de supervision des serveurs ne permettent pas de trancher seuls : **mes applications rendent-elles réellement le service attendu ?** Un serveur peut être allumé, son processeur peu chargé et son service web à l'écoute, sans que les utilisateurs puissent se connecter, rechercher un produit ou valider une commande. La sonde vérifie ces fonctions de bout en bout, comme le ferait un utilisateur ou une application cliente, et signale tout écart.

Elle exécute plusieurs familles de contrôles, selon l'édition acquise :

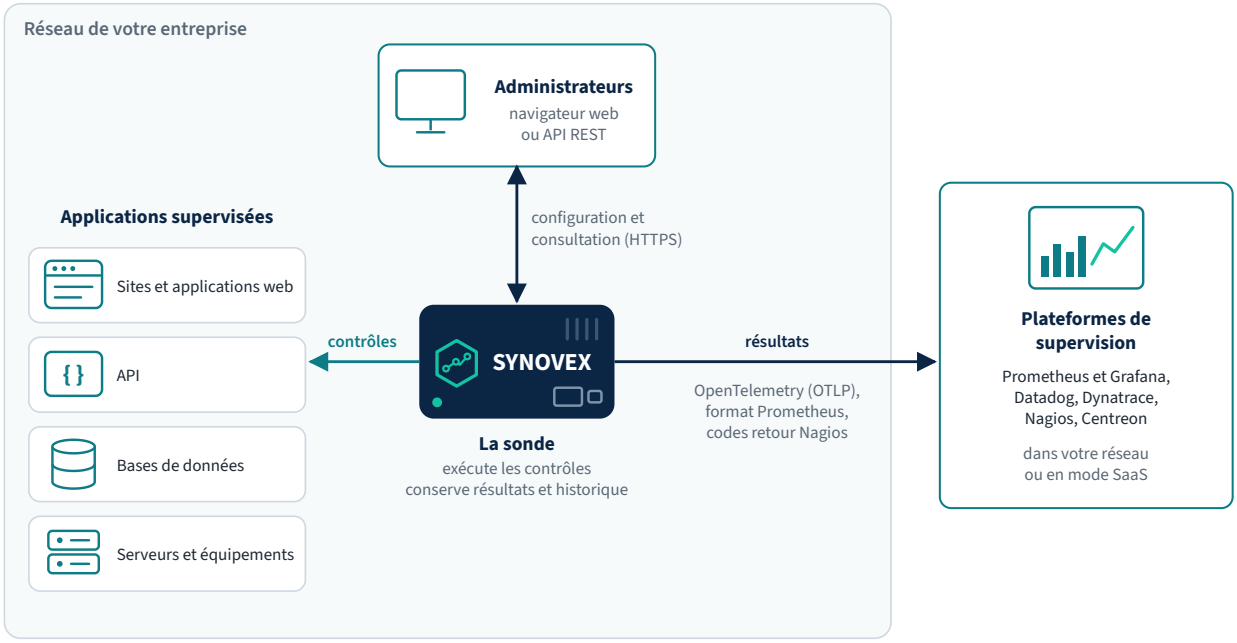
Famille	Exemples de vérifications	Chapitre de référence
Contrôles réseau et techniques	Réponse d'un équipement au ping, ouverture d'un port TCP, résolution DNS, réponse d'un site web et contenu de la page, validité d'un certificat, plugins compatibles Nagios	Chapitre 10
Contrôles applicatifs et métier	Comportement d'une API, enchaînement de requêtes, cohérence d'une réponse, requête sur une base de données, suivi d'un indicateur métier tel que le nombre de commandes du jour	Chapitre 11
Scénarios navigateur	Parcours utilisateur automatisé dans un vrai navigateur (Chromium) piloté par Puppeteer : connexion, navigation, recherche, panier, vérification d'éléments	Chapitres 13 à 16

Trois principes distinguent cette sonde d'un simple script de surveillance.

Elle est autonome. Elle planifie et exécute elle-même ses contrôles, conserve leurs résultats et leur historique, et se consulte localement depuis un navigateur web. Elle ne dépend d'aucun serveur externe pour fonctionner : une panne de votre plateforme de supervision, de votre accès Internet ou d'un service de l'éditeur n'interrompt pas les vérifications.

Elle est ouverte. Les résultats sont publiés selon des standards reconnus : OpenTelemetry et son protocole OTLP, format d'exposition Prometheus, codes retour compatibles Nagios. Ils s'intègrent ainsi aux plateformes que vous utilisez déjà (Prometheus et Grafana, Datadog, Dynatrace, Nagios, Centreon) sans connecteur propriétaire.

Elle est administrable à distance de manière sécurisée. Une interface d'administration web et une API REST documentée permettent de la configurer depuis le réseau où elle est installée ou au travers d'un accès sécurisé à ce réseau, avec des comptes nominatifs, des rôles, des jetons d'API et un journal des opérations.



Les flèches indiquent le sens de circulation des informations. Détail des flux : chapitres 3 et 7.

Figure 1-1 — Vue d'ensemble, exemple d'implantation dans le réseau de l'entreprise : la sonde, les applications supervisées et les plateformes de supervision qui reçoivent les résultats

1.2 Objectifs du manuel

Ce manuel accompagne la carte microSD ou le boîtier complet. Il vous permet de :

1. comprendre l'intérêt de la supervision applicative et la place de la sonde dans votre dispositif de supervision (partie I) ;
2. préparer, installer et mettre en service la sonde en toute sécurité (partie II) ;
3. configurer et administrer les contrôles, les planifications et les plans d'exécution (partie III) ;
4. concevoir des scénarios navigateur fiables avec Puppeteer, à partir d'exemples progressifs (partie IV) ;
5. exploiter les résultats, les intégrer à vos plateformes d'observabilité, diagnostiquer les incidents et maintenir la sonde dans la durée (partie V).

Il sert à la fois de guide de mise en service, à suivre pas à pas, et de référence technique, à consulter ponctuellement. Les annexes (chapitre 21) regroupent les éléments de référence : glossaire, questions fréquentes, liste de vérification de mise en service, tableau des flux réseau, référence de l'API et des configurations, types d'erreur et coordonnées du support.

1.3 Public concerné

Le manuel s'adresse à plusieurs profils. Chaque partie indique en introduction le niveau technique attendu.

Profil	Besoins principaux	Parties les plus utiles
Responsable infrastructure, DSI, chef de projet	Comprendre l'apport de la solution, préparer le déploiement, mobiliser les bonnes équipes	Partie I, chapitre 4, chapitre 18
Technicien chargé de l'installation	Installer physiquement la sonde, la raccorder et la démarrer	Chapitres 4, 5 et 6
Administrateur système et réseau	Configurer le réseau, la sécurité, les comptes, les mises à jour ; dépanner	Partie II, chapitres 9, 19 et 20
Équipe d'exploitation, centre de supervision	Suivre les états, interpréter les résultats, réagir aux alertes	Chapitres 8, 17 et 18
Développeur, intégrateur, équipe qualité	Créer des contrôles applicatifs et des scénarios Puppeteer	Chapitres 11 et 12, partie IV
Responsable sécurité	Évaluer les flux, l'isolation, la gestion des secrets et des accès	Chapitres 3, 7 et 9

Aucune compétence de développement n'est nécessaire pour installer la sonde, la mettre en service et configurer les contrôles réseau, HTTP ou de base de données. L'écriture de scénarios scriptés (chapitre 14) suppose en revanche une connaissance élémentaire de JavaScript. Les scénarios déclaratifs, décrits dans le même chapitre, permettent de construire des parcours simples sans programmation.

1.4 Périmètre de la solution

1.4.1 Éditions

Les fonctions disponibles dépendent de l'édition acquise : SYNOVEX Essential, SYNOVEX Professional ou SYNOVEX Enterprise. Dans les tableaux de cette documentation, les éditions sont désignées par leur seul nom (Essential, Professional, Enterprise). Toutes les éditions partagent le même système. L'édition est inscrite sur la carte microSD lors de sa préparation, sous la forme d'un **certificat d'édition** qui active les modules correspondants : aucune clé n'est à saisir et aucune activation en ligne n'est nécessaire. Le droit d'utilisation du logiciel intégré est perpétuel : il ne nécessite ni abonnement ni renouvellement. Chaque version majeure bénéficie d'une période de support d'au moins cinq ans, dont la date de fin vous est indiquée à l'achat : pendant cette période, les correctifs de sécurité sont publiés pour toutes les sondes. Un contrat de maintenance facultatif donne en outre accès aux nouvelles versions fonctionnelles ; son échéance n'affecte aucune fonction acquise (chapitre 20, § 20.4.7). Pour passer à une édition supérieure, vous remplacez la carte par une carte de cette édition et vous y transférez votre configuration (chapitre 20, § 20.6), sous réserve que le Raspberry Pi 5 corresponde aux exigences de l'édition visée (chapitre 4).

Fonction	Essential	Professional	Enterprise
Contrôles ICMP, TCP, DNS	Oui	Oui	Oui
Contrôles HTTP/HTTPS, y compris vérification du contenu et du certificat	Oui	Oui	Oui
Plugins compatibles Nagios fournis avec la sonde	Oui	Oui	Oui

Fonction	Essential	Professional	Enterprise
Plugins compatibles Nagios personnalisés	—	Oui	Oui
Contrôles d'API (suites de requêtes, extraction de valeurs)	—	Oui	Oui
Contrôles de bases de données (PostgreSQL, MySQL/MariaDB, SQL Server, Oracle) et indicateurs métier	—	Oui	Oui
Scénarios navigateur déclaratifs et scriptés (Puppeteer, Chromium headless)	—	—	Oui
Interface d'administration, API REST, planification, plans d'exécution, historique local	Oui	Oui	Oui
Export OpenTelemetry (OTLP), endpoint Prometheus, plugin Nagios/Centreon	Oui	Oui	Oui

Aucune édition n'impose de limite technique au nombre de contrôles ou de scénarios navigateur. La capacité réelle dépend du matériel, de la fréquence et de la durée des contrôles : la méthode de dimensionnement (calcul de la charge, seuils à respecter, surveillance dans la durée) figure au chapitre 12 (§ 12.6), et l'annexe C (chapitre 21, § 21.3) récapitule, sous forme d'aide-mémoire, les prérequis et la configuration matérielle minimale.

1.4.2 Offres matérielles

La sonde est proposée sous deux formes :

Offre	Contenu	Ce que vous fournissez
microSD préconfigurée	Carte microSD contenant le système et le logiciel de la sonde	Raspberry Pi 5 de capacité mémoire adaptée à l'édition, alimentation, refroidissement, boîtier, câble Ethernet
Boîtier complet	Raspberry Pi 5 assemblé dans son boîtier, refroidissement, carte microSD installée	Alimentation et câble Ethernet, sauf mention contraire de la liste de colisage jointe au boîtier et de votre bon de commande ; emplacement d'installation

Un SSD NVMe facultatif, monté sur la carte d'extension M.2 HAT+ officielle de Raspberry Pi, peut compléter la carte microSD comme volume de données. Il est surtout utile pour l'édition Enterprise ; la carte microSD reste le support principal et suffit à un usage standard (chapitre 20, § 20.3.4).

Le contenu exact de chaque offre et les références de matériel sont détaillés au chapitre 4.

> **Information** — SYNOVEX est conçu et documenté pour Raspberry Pi 5. Son architecture logicielle peut permettre une utilisation sur d'autres ordinateurs monocartes, notamment certains modèles Orange Pi. Toutefois, ces plateformes n'ont pas fait l'objet d'une validation officielle et leur fonctionnement n'est pas garanti. SYNOVEX est pris en charge exclusivement sur **Raspberry Pi 5** raccordé en **Ethernet** (chapitre 4, § 4.1.5).

1.4.3 Ce que la sonde fait et ne fait pas

La sonde **observe depuis l'extérieur** le comportement des services, à la manière d'un utilisateur ou d'une application cliente. Elle n'installe aucun agent sur les serveurs supervisés et n'a pas besoin d'accès privilégié à ces serveurs. En contrepartie, elle ne mesure pas l'utilisation du processeur, de la mémoire ou des disques des serveurs : cette supervision des ressources reste du ressort de vos outils existants, que la sonde complète sans les remplacer (chapitre 2).

La sonde n'est pas non plus un outil de test de charge. Ses contrôles sont volontairement peu fréquents et légers, pour ne pas perturber les services observés. Elle n'est pas davantage un outil de mesure de l'expérience des utilisateurs réels : elle mesure des parcours représentatifs, reproduits à l'identique, ce qui la rend complémentaire des solutions de mesure en conditions réelles (chapitre 2, § 2.6.3).

1.4.4 Évolutions prévues

Les évolutions suivantes sont envisagées pour des versions ultérieures. Elles ne font pas partie des fonctions actuelles de la sonde : ne les prenez pas en compte pour préparer un déploiement.

- **Console centrale.** Une console centrale facultative permettra de gérer plusieurs sondes depuis un point unique. Elle sera hébergée par l'éditeur ou installée dans votre infrastructure, au choix. La sonde restera pleinement autonome : la console ne sera jamais nécessaire à son fonctionnement. L'adresse de la console hébergée par l'éditeur est réservée (`https://fleet-synovex.incitypark.fr`) ; la sonde ne l'utilise pas.
- **Connexion Wi-Fi.** La sonde se raccorde aujourd'hui exclusivement en Ethernet ; une connexion Wi-Fi est à l'étude.

1.5 Organisation du manuel

Le manuel suit la progression naturelle d'un projet de supervision : comprendre, installer, configurer, utiliser, diagnostiquer.

Partie	Chapitres	Contenu
I — Présentation de la solution	1 à 3	Introduction, enjeux de la supervision applicative, architecture
II — Installation et mise en service	4 à 7	Prérequis, installation physique, premier démarrage, réseau et sécurité
III — Utilisation de la sonde	8 à 12	Principes de fonctionnement, administration, contrôles techniques, contrôles applicatifs, planification
IV — Scénarios Puppeteer	13 à 16	Introduction à Puppeteer, construction d'un scénario, exemples progressifs, bonnes pratiques
V — Exploitation et maintenance	17 à 20	Résultats, intégrations aux plateformes d'observabilité, dépannage, maintenance
Annexes	21	Glossaire, FAQ, listes de vérification, références

1.6 Statut du document

Ce manuel décrit le fonctionnement de référence de **SYNOVEX** sur Raspberry Pi 5. Les valeurs qui dépendent de chaque version de l'image système, comme les versions des composants embarqués ou les valeurs mesurées sur le matériel, ne sont pas reproduites dans le texte : elles figurent dans les notes de version. Le document applique les conventions suivantes.

Marquage	Signification	Conduite à tenir
Texte sans marquage	Fonctionnement de référence	Suivre comme fonctionnement de référence ; pour la version installée sur votre sonde, consulter aussi ses notes de version
Renvoi aux notes de version ou à `GET /api/v1/system`	Valeur propre à chaque version de l'image système : version d'un composant (champ `versions`), valeur mesurée sur Raspberry Pi 5, précision de mise en œuvre	Se reporter aux notes de version de la version installée, ou interroger la sonde (chapitre 9)
Valeur donnée « à titre indicatif »	Ordre de grandeur, qui dépend de la configuration et de la charge de la sonde	Ne pas dimensionner au plus juste sur cette valeur ; surveiller la sonde dans la durée (chapitre 12, § 12.6)

Marquage	Signification	Conduite à tenir
Exemple (`example.com` , `192.0.2.0/24` , comptes de test)	Valeur fictive, utilisée à titre d'illustration (§ 1.7.4)	Remplacer systématiquement par vos propres valeurs
Encadré **Évolution prévue**	Fonction prévue dans une version ultérieure	Ne pas planifier son usage tant qu'elle n'est pas annoncée comme disponible

> ! **Attention** — Ce manuel décrit le fonctionnement du produit ; il ne constitue ni un rapport d'essai, ni une certification, ni un engagement de performance. Avant de fonder un déploiement sur une fonction qui conditionne la disponibilité ou la sécurité (mise à jour A/B, chien de garde, SSD, 802.1X, IPv6, liaison à la carte, mémoire de Chromium, intégrations, alertes), consultez les notes de version de la version installée et vérifiez le comportement sur votre sonde lors de la mise en service (liste de vérification de l'annexe D).

La version du manuel et la version du logiciel auxquelles il correspond figurent dans les informations de version (chapitre 21). La version logicielle installée sur votre sonde s'affiche dans l'interface d'administration et via l'API (chapitre 9). En cas d'écart entre ce manuel et le comportement observé, le comportement de la sonde et les notes de version priment ; signalez l'écart au support pour correction du document.

> i **Information** — Les identifiants techniques de la sonde (préfixe `sva` des métriques, services `sva-agent` et `sva-runner` , nom d'hôte par défaut `sva-XXXXXX` , plugin `check_sva`) sont **stables** et indépendants du nom commercial du produit, **SYNOVEX**. Ils proviennent de la désignation technique interne « SVA » et ne changeront pas si le nom commercial évolue.

1.7 Conventions de lecture

1.7.1 Typographie

Convention	Usage	Exemple
`Police à chasse fixe`	Commandes, noms de fichiers, valeurs à saisir, extraits de configuration, champs d'API, états	`provision.yaml` , `CRITICAL` , `timeout`
Gras	Libellés importants, termes définis, éléments d'interface	**Code d'installation**
Italique	Termes étrangers non traduits, mises en relief ponctuelles	*headless*
`<texte entre chevrons>`	Valeur à remplacer par la vôtre	`https://<adresse-de-la-sonde>/`
« § 7.4 »	Renvoi à une section du manuel	voir § 7.4

Les états d'un contrôle (`OK` , `WARNING` , `CRITICAL` , `UNKNOWN` , `STALE`) sont toujours écrits en majuscules et en police à chasse fixe, car ils correspondent à des valeurs exactes visibles dans l'interface, l'API et les plateformes d'observabilité.

1.7.2 Encadrés

Cinq types d'encadrés signalent des informations particulières :

> i **Information** — Complément utile à la compréhension, sans conséquence directe sur l'opération en cours.

✓ **Conseil** — Bonne pratique recommandée par l'éditeur.

! **Attention** — Risque de dysfonctionnement, de résultat erroné ou de perte de données si la consigne n'est pas respectée.

! **Avertissement** — Risque pour la sécurité de votre système d'information, pour le matériel ou pour les personnes. Lisez systématiquement ces encadrés avant d'agir.

→ **Évolution prévue** — Fonction annoncée pour une version ultérieure, non disponible actuellement.

1.7.3 Procédures

Les procédures sont présentées selon une structure constante, pour que vous puissiez vérifier chaque étape avant de passer à la suivante :

Rubrique	Contenu
Objectif	Ce que la procédure permet d'obtenir
Prérequis	Conditions à réunir avant de commencer : matériel, droits, informations
Étapes	Actions numérotées, à réaliser dans l'ordre
Résultat attendu	État dans lequel la sonde ou le système doit se trouver à l'issue
Vérification	Moyen de contrôler objectivement ce résultat
En cas de problème	Renvoi vers la section de dépannage correspondante (chapitre 19)

Lorsqu'une étape présente un risque, l'encadré **Attention** ou **Avertissement** est placé **avant** l'étape concernée, et non après.

1.7.4 Exemples fictifs

Tous les exemples du manuel sont fictifs. Ils utilisent des noms de domaine réservés à la documentation (`example.com`, `example.org`, `example.net`, et `example.internal` pour les noms internes), des adresses IP réservées à la documentation (plages `192.0.2.0/24`, `198.51.100.0/24` et `203.0.113.0/24`) et des comptes de test fictifs tels que `test-supervision@example.com`. Ces valeurs ne correspondent à aucun service réel : remplacez-les systématiquement par vos propres valeurs.

! **Avertissement** — N'utilisez jamais de comptes personnels ou de données réelles de clients dans un contrôle ou un scénario. Créez des comptes de test dédiés, aux droits limités, comme décrit au chapitre 4 (§ 4.6).

1.7.5 Illustrations

Les figures sont numérotées d'après le chapitre, suivi d'un numéro d'ordre. Les représentations du matériel sont schématiques et ne sont pas à l'échelle. Les opérations réalisées dans l'interface d'administration sont décrites dans le texte et, lorsque c'est utile, par la requête équivalente de l'API REST (chapitre 9) ; le manuel ne reproduit pas de capture de cette interface.

1.8 Parcours de lecture conseillés

Le manuel peut se lire de façon linéaire, mais la plupart des lecteurs gagneront à suivre le parcours correspondant à leur rôle.

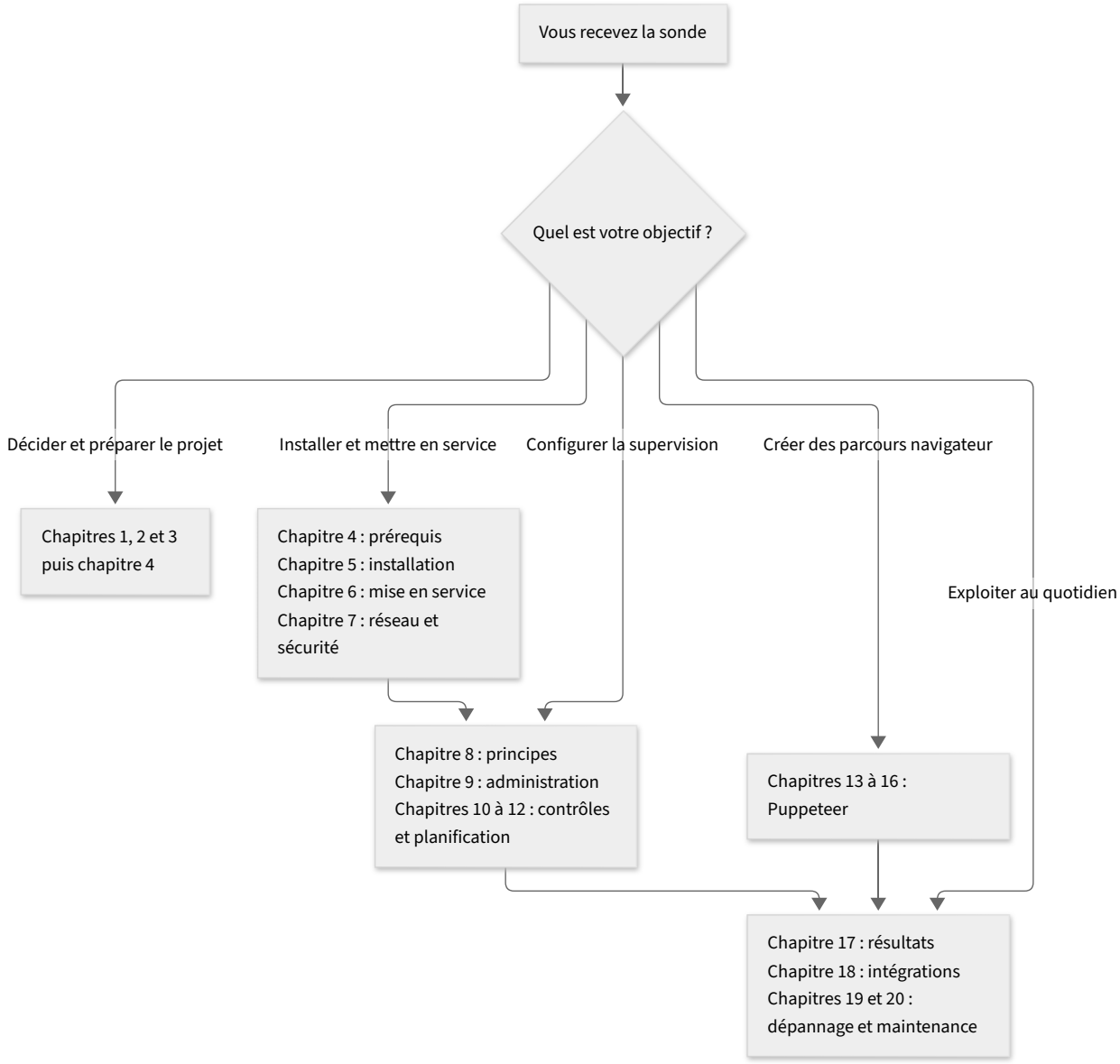


Schéma 1-1 — Parcours de lecture selon votre rôle

Mise en service rapide. Pour une première installation, suivez dans l'ordre la liste de vérification avant démarrage (§ 4.9), l'installation physique (chapitre 5) et la mise en service (chapitre 6), puis créez un premier contrôle HTTP simple (chapitre 10). Une fois les prérequis réunis, la mise en service se limite à ces étapes ; la durée d'un déploiement dépend surtout de la préparation décrite ci-dessous.

Préparation d'un projet. Avant même de recevoir la sonde, transmettez le chapitre 4 et le tableau des flux réseau (chapitre 21) à vos équipes réseau et sécurité. Les ouvertures de flux et la création des comptes de test sont généralement les étapes les plus longues d'un déploiement.

Création de scénarios. Les développeurs et les équipes qualité peuvent commencer par le chapitre 13, puis travailler les exemples du chapitre 15 sur un poste de travail, avec le lanceur local fourni dans les exemples, avant de les déployer sur la sonde.

1.9 Support et retour d'expérience

Pour toute question non couverte par ce manuel, consultez d'abord la FAQ et les listes de vérification du chapitre 21, puis le chapitre 19 consacré au dépannage. Si le problème persiste, contactez le support depuis le portail `https://support-synovex.incitypark.fr`, dès l'annonce de sa mise en service ; d'ici là, utilisez le canal de contact convenu dans votre contrat. Les autres coordonnées et les horaires figurent à l'annexe L (§ 21.12), selon les conditions de votre contrat. Préparez alors la version logicielle de la sonde, son édition, la description précise du symptôme et, si possible, l'identifiant de l'exécution concernée (chapitre 17).

Vos remarques sur ce manuel sont les bienvenues : transmettez-les au support, par le portail ou par le canal convenu dans votre contrat.

Adresses de l'éditeur. Le plan de nommage officiel est le suivant ; les services correspondants sont mis en service progressivement, et la disponibilité de chacun est annoncée par l'éditeur, dans les notes de version ou par le canal convenu dans votre contrat. Une procédure de ce manuel qui s'appuie sur l'un de ces services ne s'applique qu'après cette annonce (annexe L, § 21.12) :

Adresse	Fonction
`https://synovex.incitypark.fr`	Site commercial
`https://docs-synovex.incitypark.fr`	Documentation technique en ligne et notes de version
`https://updates-synovex.incitypark.fr`	Distribution des mises à jour
`https://support-synovex.incitypark.fr`	Support et assistance
`https://fleet-synovex.incitypark.fr`	Console de gestion centralisée des sondes, prévue dans une version ultérieure

La sonde n'a besoin d'aucune de ces adresses pour exécuter ses contrôles. Seul le service de distribution des mises à jour est contacté par la sonde elle-même, pour les mises à jour en ligne (chapitre 20) ; les adresses qu'elle utilise proviennent de son profil éditeur signé.

Chapitre 2 — Comprendre la supervision applicative

Ce chapitre présente les notions sur lesquelles repose la sonde. Il ne suppose aucune connaissance préalable de la supervision et peut être lu par des interlocuteurs non techniques : responsables métier, chefs de projet, direction. Les lecteurs familiers des outils de supervision peuvent se limiter aux § 2.4 à § 2.8.

2.1 Qu'est-ce que la supervision ?

La **supervision** (en anglais *monitoring*) désigne l'ensemble des moyens qui permettent de connaître en permanence l'état d'un système informatique et d'être averti lorsqu'il s'écarte de son fonctionnement normal. Elle repose sur trois activités complémentaires :

1. **Mesurer** : collecter régulièrement des informations sur les composants et les services (un serveur répond-il ? en combien de temps ? une page contient-elle le texte attendu ?).
2. **Évaluer** : comparer ces mesures à des critères définis à l'avance (seuils, résultats attendus) pour en déduire un **état** : normal, dégradé ou en panne.
3. **Alerter et conserver** : prévenir les bonnes personnes lorsqu'un état se dégrade, et garder l'historique pour analyser les incidents, mesurer la qualité de service et anticiper les problèmes.

Le terme **observabilité**, apparu plus récemment, désigne la capacité à comprendre le comportement interne d'un système à partir des données qu'il produit : métriques, traces et journaux. La supervision répond à la question « quelque chose ne va-t-il pas ? ». L'observabilité aide à répondre à « pourquoi ? ». La sonde contribue aux deux : elle détecte les dysfonctionnements et fournit les éléments qui permettent d'en comprendre la cause (étape en échec, type d'erreur, capture d'écran, durées détaillées).

2.2 Disponibilité, performance et fiabilité

Trois notions structurent toute démarche de supervision. Elles sont souvent confondues ; les distinguer permet de choisir les bons contrôles.

Notion	Question posée	Exemple de mesure	Exemple de dysfonctionnement
Disponibilité	Le service est-il accessible et rend-il le service attendu ?	Pourcentage d'exécutions réussies sur une période	Le site renvoie une erreur ; la connexion des utilisateurs échoue
Performance	Le service répond-il dans un délai acceptable ?	Temps de réponse d'une page, durée d'un parcours, temps par étape	La recherche de produits prend 12 secondes au lieu de 2
Fiabilité	Le service se comporte-t-il de façon constante dans le temps ?	Fréquence des échecs intermittents, variabilité des temps de réponse	Une erreur sur dix tentatives, imprévisible

La disponibilité s'exprime généralement en pourcentage sur une période. Le tableau suivant montre ce que représentent concrètement quelques objectifs courants :

Disponibilité visée	Indisponibilité tolérée par mois (30 jours)	Indisponibilité tolérée par an
99 %	7 h 12 min	3 j 15 h 36 min

Disponibilité visée	Indisponibilité tolérée par mois (30 jours)	Indisponibilité tolérée par an
99,5 %	3 h 36 min	1 j 19 h 48 min
99,9 %	43 min 12 s	8 h 45 min 36 s
99,99 %	4 min 19 s	52 min 34 s

Ces objectifs n'ont de sens que si la disponibilité est mesurée **du point de vue de l'utilisateur**. Un serveur disponible à 99,99 % ne garantit rien si l'application qu'il héberge est inutilisable une demi-journée par mois. C'est précisément ce que la supervision applicative permet de mesurer.

> **Information** — Un objectif de niveau de service se formalise souvent par un **SLO** (*Service Level Objective*), par exemple « 99,5 % des parcours de commande réussissent en moins de 8 secondes sur un mois glissant ». Les résultats de la sonde, exportés vers votre plateforme d'observabilité (chapitre 18), permettent de calculer ces indicateurs.

2.3 Les niveaux de supervision

On distingue habituellement plusieurs niveaux de supervision, du plus matériel au plus fonctionnel. Chacun répond à des questions différentes et s'appuie sur des outils différents.

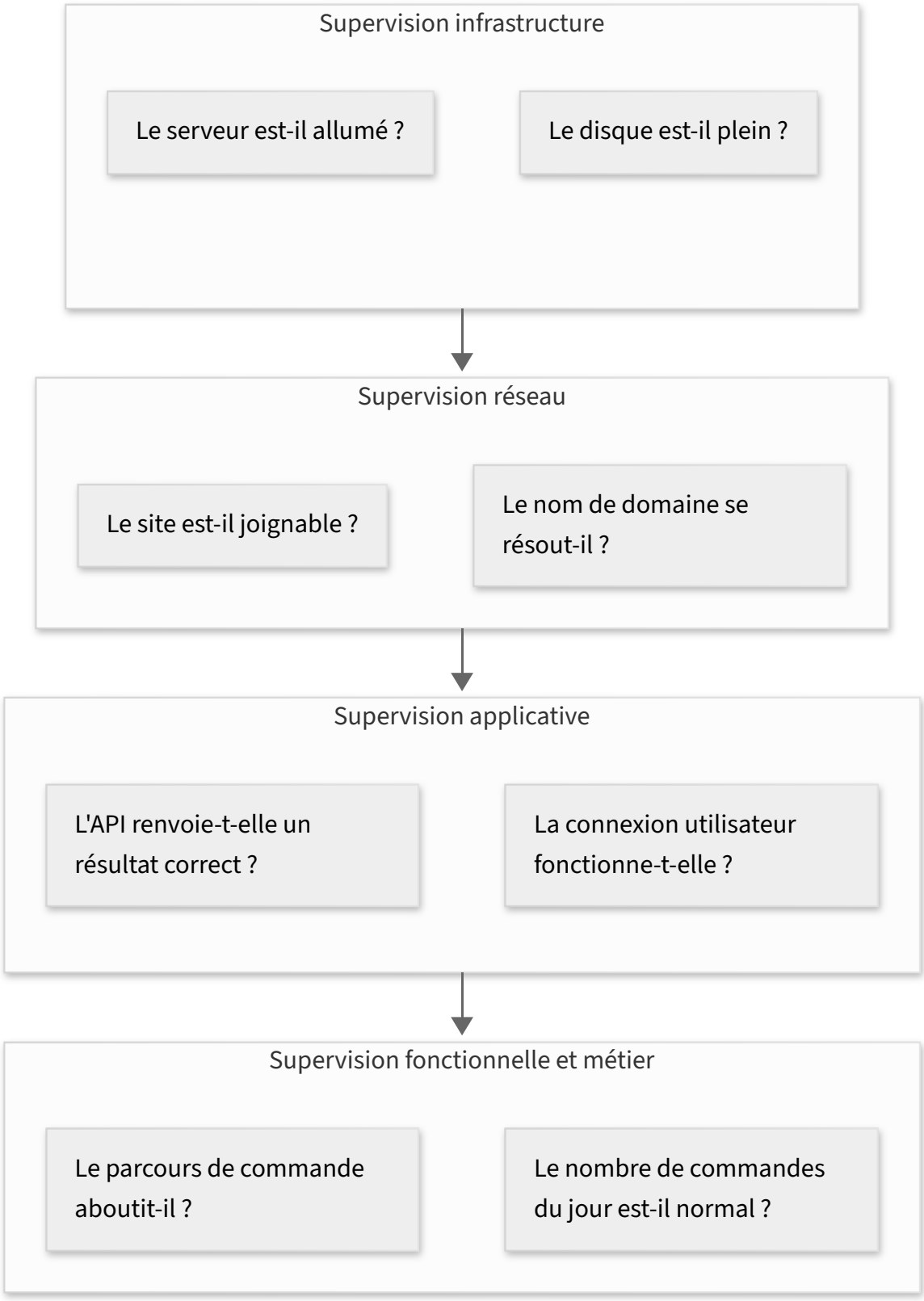


Schéma 2-1 — Les niveaux de supervision, du matériel au métier

2.3.1 Supervision de l'infrastructure

Elle surveille les ressources matérielles et système : processeur, mémoire, disques, température, état des services du système d'exploitation, sauvegardes. Elle est généralement assurée par des agents installés sur les serveurs ou par des protocoles d'administration (SNMP, WMI) et par des outils tels que Nagios, Centreon ou Zabbix. Elle est indispensable pour anticiper la saturation d'une ressource et diagnostiquer une panne matérielle.

2.3.2 Supervision du réseau

Elle vérifie que les équipements et les services sont joignables et que les communications s'établissent : réponse au ping, ouverture des ports, résolution des noms, latence, perte de paquets. Elle détecte les coupures de liens, les erreurs de routage ou de filtrage et les défaillances de DNS.

2.3.3 Supervision applicative

Elle vérifie que les applications rendent le service attendu, en les sollicitant comme le ferait un utilisateur ou une application cliente : requête vers une API et analyse de la réponse, ouverture d'une page web et vérification de son contenu, connexion avec un compte de test, requête sur une base de données. Elle ne se contente pas de constater qu'un service « répond » : elle vérifie qu'il répond **correctement**.

2.3.4 Supervision fonctionnelle et métier

Elle va plus loin en vérifiant des enchaînements complets et des indicateurs propres à l'activité : un parcours d'achat aboutit-il ? le nombre de commandes enregistrées depuis le début de la journée est-il cohérent avec l'activité habituelle ? un traitement nocturne a-t-il produit les données attendues ? Elle traduit la santé du système d'information dans les termes du métier.

La sonde couvre les niveaux réseau, applicatif et fonctionnel. Elle s'appuie sur la supervision de l'infrastructure existante sans s'y substituer.

2.4 Pourquoi un serveur disponible ne garantit pas le bon fonctionnement d'une application

Une application moderne repose sur de nombreux composants : serveurs web, serveurs d'application, bases de données, files de messages, services tiers (paiement, authentification, cartographie), réseaux de diffusion de contenu, scripts exécutés dans le navigateur. Chacun peut fonctionner individuellement alors que l'ensemble ne rend plus le service. Les situations suivantes, toutes rencontrées en exploitation, illustrent cet écart. Elles sont fictives dans leurs détails.

2.4.1 Le site fonctionne, mais le bouton de commande est défaillant

Une boutique en ligne s'affiche normalement. Les serveurs web répondent, leur processeur est peu chargé, la base de données est disponible : la supervision de l'infrastructure ne signale rien. Pourtant, après une mise à jour, un script exécuté dans le navigateur provoque une erreur au moment où l'utilisateur clique sur **Valider ma commande**. Le bouton ne réagit plus. Aucune commande n'aboutit, et l'incident n'est découvert que plusieurs heures plus tard, par les réclamations des clients ou par l'équipe commerciale qui constate l'absence de ventes.

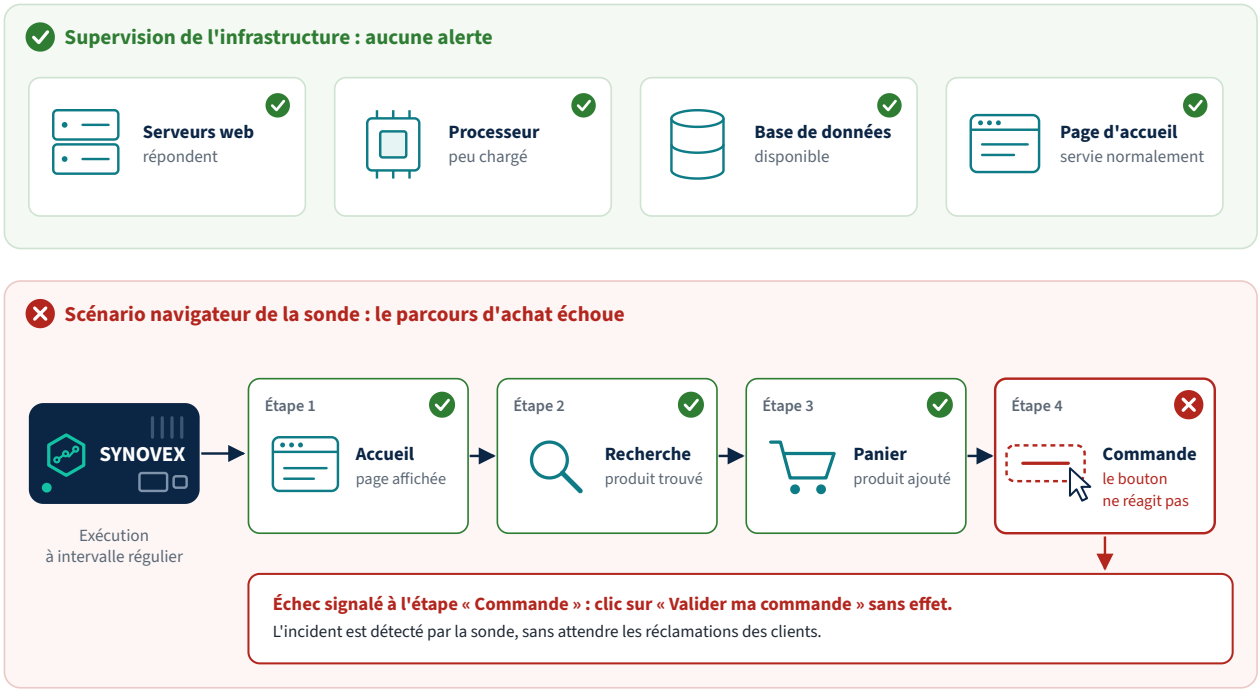


Figure 2-1 — Exemple : l'infrastructure ne signale aucune anomalie, mais le parcours d'achat échoue à l'étape de commande ; le scénario de la sonde le détecte

Seul un contrôle qui **reproduit le parcours d'achat dans un vrai navigateur**, jusqu'au clic sur ce bouton, peut détecter ce type de défaut. C'est le rôle des scénarios navigateur (partie IV).

2.4.2 L'API répond HTTP 200, mais le résultat est faux

Une API de calcul de prix répond avec le code HTTP `200 OK`, qui signifie « requête traitée avec succès ». Un contrôle qui se limite au code de retour la considère donc comme disponible. Or, à la suite d'une erreur de configuration, l'API renvoie un prix à zéro pour tous les articles, ou une liste vide au lieu de la liste des produits disponibles.

Un contrôle applicatif doit donc **vérifier le contenu de la réponse** : présence d'un champ, valeur attendue, nombre d'éléments, cohérence d'un montant. Les contrôles HTTP et API de la sonde permettent d'exprimer ces vérifications, appelées **assertions** (chapitres 10 et 11).

2.4.3 La base de données répond, mais les traitements métier échouent

Le serveur de base de données accepte les connexions et répond aux requêtes simples. Mais un traitement planifié chargé d'intégrer les commandes de la nuit a échoué silencieusement : la table des commandes à préparer est vide ce matin. Les entrepôts n'expédient rien, et la supervision technique reste au vert.

Une requête de contrôle qui vérifie qu'**au moins un enregistrement a été intégré depuis minuit**, ou que la date de la dernière intégration est récente, signale immédiatement l'anomalie. Les contrôles de bases de données de la sonde (chapitre 11) reposent sur ce principe.

2.4.4 Le site est accessible, mais l'authentification ne fonctionne plus

La page de connexion d'une application s'affiche. Mais l'annuaire d'entreprise ou le fournisseur d'identité utilisé pour vérifier les mots de passe est inaccessible depuis le serveur d'application, ou son certificat a expiré. Tous les utilisateurs reçoivent le message « identifiant ou mot de passe incorrect ». Le service est, de fait, totalement indisponible, alors que tous les contrôles techniques sont positifs.

Un scénario qui **se connecte avec un compte de test** et vérifie l'affichage de l'espace personnel détecte ce dysfonctionnement en quelques minutes.

2.4.5 Le nombre de commandes chute anormalement

Aucune erreur technique n'est visible, mais le nombre de commandes enregistrées entre 10 h et 11 h est cinq fois inférieur à celui des semaines précédentes à la même heure. Les causes possibles sont multiples : un moyen de paiement refusé pour une catégorie de cartes, une page produit qui ne s'affiche plus sur certains navigateurs, une campagne de communication renvoyant vers une adresse erronée.

La surveillance d'un **indicateur métier**, le volume de commandes sur une période, avec un seuil minimal adapté à l'heure de la journée, révèle des problèmes qu'aucun contrôle technique ne peut voir. La sonde permet de suivre ces indicateurs par des requêtes sur les bases de données et de les publier comme métriques (chapitre 11).

2.4.6 L'application fonctionne techniquement, mais le parcours devient inutilisable

Toutes les pages s'affichent et tous les services répondent. Mais une modification de la mise en page fait qu'une bannière d'information recouvre le bouton **Suivant** du formulaire d'inscription, ou que la page de résultats de recherche met vingt secondes à s'afficher. Techniquement, rien n'est « en panne ». Pour l'utilisateur, le service est inutilisable.

Un scénario navigateur qui **mesure la durée de chaque étape** et échoue si un élément n'est pas cliquable signale ces dégradations, en précisant l'étape concernée.

2.4.7 Synthèse

Situation	Supervision infrastructure	Supervision réseau	Contrôle HTTP simple (code retour)	Supervision applicative et fonctionnelle
Bouton de commande défaillant	Ne détecte pas	Ne détecte pas	Ne détecte pas	Détecte (scénario navigateur)
API 200 avec résultat faux	Ne détecte pas	Ne détecte pas	Ne détecte pas	Détecte (assertion sur le contenu)
Traitement métier en échec	Ne détecte pas	Ne détecte pas	Ne détecte pas	Détecte (requête de contrôle)
Authentification cassée	Ne détecte pas	Ne détecte pas	Ne détecte pas	Détecte (scénario de connexion)
Chute des commandes	Ne détecte pas	Ne détecte pas	Ne détecte pas	Détecte (indicateur métier)
Parcours inutilisable ou très lent	Ne détecte pas	Ne détecte pas	Partiellement	Détecte (mesure par étape)
Serveur arrêté	Détecte	Détecte	Détecte	Détecte
Disque saturé, avant impact	Détecte	Ne détecte pas	Ne détecte pas	Ne détecte pas

La dernière ligne rappelle que la supervision de l'infrastructure reste indispensable : elle détecte les risques **avant** qu'ils aient un impact visible. Les deux approches se complètent.

2.5 Limites des contrôles techniques classiques

Les contrôles techniques classiques présentent quatre limites que la supervision applicative compense.

Ils observent les composants, pas le service. Un contrôle porte sur un serveur, un processus ou un port. Le service rendu à l'utilisateur résulte de la coopération de dizaines de composants, dont certains échappent à votre supervision : services tiers, réseaux de diffusion, scripts exécutés dans le navigateur.

Ils se limitent souvent à la présence d'une réponse. Un port ouvert, un processus actif ou un code HTTP `200` indiquent que le composant répond, pas qu'il répond correctement.

Ils sont vus depuis l'intérieur. Un agent installé sur un serveur ne voit pas le chemin parcouru par l'utilisateur : résolution DNS, pare-feu, proxy, équilibreur de charge, certificat présenté au navigateur. Une erreur sur ce chemin reste invisible.

Ils ignorent la dimension métier. Un contrôle technique ne sait pas qu'une journée sans commande est anormale, ni qu'un délai de réponse acceptable pour un traitement de fond est inacceptable pour un client sur une page de paiement.

2.6 L'intérêt de la supervision synthétique

2.6.1 Définition

La **supervision synthétique** consiste à exécuter, à intervalles réguliers et de façon automatisée, des transactions **artificielles** et reproductibles qui imitent l'usage réel d'un service : chargement d'une page, appel d'une API, parcours utilisateur complet. Le terme « synthétique » signifie que ces transactions sont générées par l'outil de supervision et non par de vrais utilisateurs.

Ses propriétés font sa valeur :

Propriété	Bénéfice
Régularité	Le service est vérifié en permanence, y compris la nuit, le week-end et pendant les périodes creuses, quand aucun utilisateur ne pourrait signaler un problème
Reproductibilité	Le même parcours est exécuté à l'identique : toute variation de résultat ou de durée provient du service, et non d'un comportement différent de l'utilisateur
Objectivité	Les mesures sont comparables dans le temps, d'une version applicative à l'autre, d'un site à l'autre
Précision du diagnostic	En cas d'échec, l'étape fautive, le message d'erreur et une capture d'écran sont disponibles immédiatement
Indépendance	Le contrôle ne dépend ni du volume de trafic ni de la bonne volonté des utilisateurs à signaler les problèmes
Choix du point d'observation	Le contrôle est exécuté depuis l'emplacement où la sonde est installée : réseau interne, site distant, réseau d'un client ou d'un partenaire, autre région géographique

Lorsque plusieurs sondes sont installées dans des environnements différents, on parle de **supervision synthétique distribuée**. Elle permet de distinguer quatre questions complémentaires :

1. **la disponibilité technique du service** : le service répond-il correctement ?
2. **la disponibilité perçue depuis un point distant** : le service est-il accessible depuis le réseau d'un client, d'une agence ou d'un partenaire ?
3. **la performance observée** : quels sont les temps de réponse, la latence et les éventuelles dégradations depuis ce point d'observation ?
4. **le fonctionnement applicatif** : les contrôles configurés confirment-ils le comportement attendu des services et des parcours applicatifs ?

Un service peut répondre correctement depuis le centre de données tout en restant inaccessible ou lent depuis une agence, un pays ou le réseau d'un client : seule une mesure prise depuis ce point d'observation le révèle. Chaque sonde rend compte de ce qu'elle observe ; la comparaison des résultats entre sondes et l'analyse des causes restent du ressort de vos équipes et de vos outils de supervision (§ 3.2.1).

2.6.2 Navigation automatisée

Pour les applications web, la forme la plus complète de supervision synthétique pilote un **véritable navigateur**. La sonde utilise Chromium, le projet de navigateur libre sur lequel repose Google Chrome, en mode **headless**, c'est-à-dire sans affichage à l'écran. Elle le pilote avec **Puppeteer**, une bibliothèque qui permet de commander le navigateur par programme : ouvrir une page, remplir un formulaire, cliquer, attendre l'apparition d'un élément, vérifier un texte.

Contrairement à une simple requête HTTP, le navigateur exécute tout ce que la page contient : scripts, chargements différés, appels d'API effectués depuis la page, redirections d'authentification. Il voit donc ce que voit l'utilisateur. Le chapitre 13 détaille ce fonctionnement et ses limites.

2.6.3 Supervision synthétique et mesure de l'expérience réelle

La supervision synthétique est souvent comparée à la **mesure de l'expérience des utilisateurs réels** (en anglais *Real User Monitoring*, RUM), qui collecte des mesures dans le navigateur de chaque visiteur.

Critère	Supervision synthétique (la sonde)	Mesure de l'expérience réelle (RUM)
Source des mesures	Transactions générées par la sonde	Navigation des utilisateurs réels
Fonctionne sans trafic	Oui	Non
Détecte un problème avant les utilisateurs	Oui, si le parcours concerné est contrôlé	Non : il faut que des utilisateurs soient touchés
Représentativité	Parcours choisis, conditions constantes	Diversité réelle des appareils, réseaux et usages
Couverture des applications internes	Oui, depuis tout emplacement où la sonde peut joindre l'application	Dépend de l'instrumentation des applications
Diagnostic d'un échec	Étape, erreur, capture d'écran	Statistiques agrégées, erreurs remontées par les navigateurs
Mise en œuvre	Sans modification des applications	Instrumentation des pages à prévoir

Les deux approches sont complémentaires. La sonde vous alerte au plus tôt sur les parcours critiques, y compris en l'absence de trafic. La mesure réelle révèle des problèmes propres à certains utilisateurs que les parcours synthétiques ne couvrent pas.

2.7 Détection proactive et réduction du temps de détection

2.7.1 Les temps d'un incident

Le déroulement d'un incident se décompose en plusieurs durées successives, qui forment ensemble la durée totale

d'impact pour les utilisateurs :

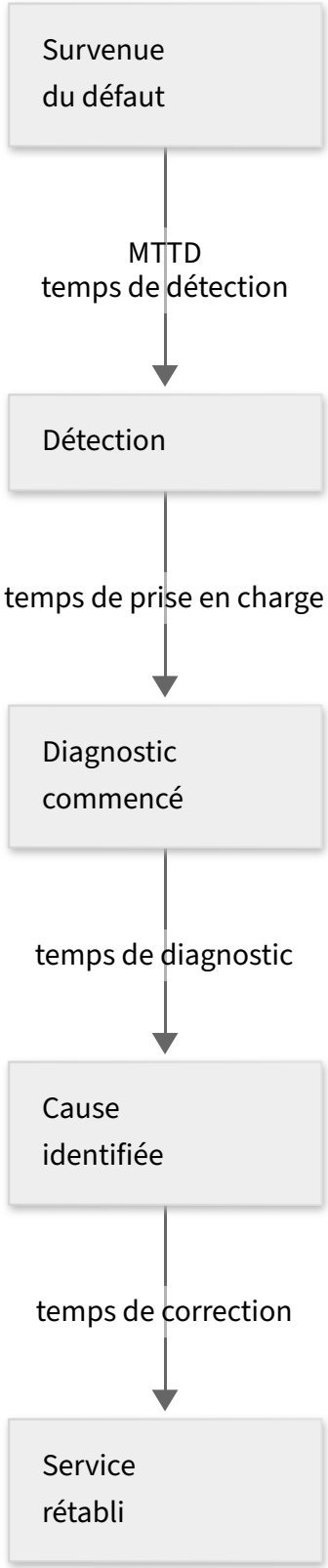


Schéma 2-2 — Les temps d'un incident : détection, diagnostic, rétablissement

Le **MTTD** (*Mean Time To Detect*, temps moyen de détection) mesure le délai entre la survenue d'un défaut et sa détection. Le **MTTR** (*Mean Time To Restore* ou *To Repair*, temps moyen de rétablissement) mesure le délai jusqu'au retour à la normale. Sans supervision applicative, la détection dépend souvent des signalements des utilisateurs : elle

peut prendre plusieurs heures, voire un jour entier pour un défaut survenu la nuit ou touchant une fonction peu utilisée.

2.7.2 Ce qui détermine le temps de détection avec la sonde

Avec la sonde, le temps de détection dépend principalement de trois paramètres que vous maîtrisez :

Paramètre	Effet	Où le régler
Fréquence d'exécution du contrôle	Un contrôle exécuté toutes les 5 minutes détecte un défaut en 5 minutes au plus, en moyenne 2 min 30 s	Chapitre 12
Nouvelles tentatives avant confirmation	Elles évitent les fausses alertes dues à un incident passager, mais retardent la confirmation d'un vrai défaut	Chapitre 8
Délai de notification de votre plateforme d'alerte	Règles d'alerte, regroupement, astreinte	Chapitre 18

Par exemple, un parcours exécuté toutes les 5 minutes avec une nouvelle tentative après 1 minute confirme un défaut persistant entre 1 et 6 minutes environ après sa survenue (hors durée d'exécution du contrôle), au lieu de plusieurs heures en l'absence de contrôle.

2.7.3 Réduction du temps de diagnostic

La sonde réduit aussi le temps de diagnostic, souvent plus long que la détection. Lorsqu'un parcours échoue, elle indique l'étape concernée, le type d'erreur (élément introuvable, délai dépassé, erreur de certificat, assertion non vérifiée…), le message détaillé, la durée de chaque étape et, pour les scénarios navigateur, une capture d'écran de la page au moment de l'échec. L'équipe d'exploitation sait immédiatement s'il s'agit d'un problème réseau, applicatif ou fonctionnel (chapitre 8) et peut solliciter directement l'équipe compétente.

2.8 Une supervision complémentaire, pas un remplacement

La supervision applicative **complète** les dispositifs existants. Le tableau suivant situe la sonde par rapport aux autres familles d'outils couramment présentes dans les entreprises.

Famille d'outils	Exemples	Ce qu'elle apporte	Ce que la sonde ajoute
Supervision d'infrastructure et réseau	Nagios, Centreon, Zabbix, outils SNMP	État des serveurs, ressources, équipements, services système	Vision du service rendu, parcours, contrôle du contenu ; plugins compatibles Nagios et intégration à ces outils (chapitre 18)
Supervision des performances applicatives (APM)	Dynatrace, Datadog APM, autres outils à agents	Analyse détaillée du code serveur, des transactions et des dépendances internes, à partir du trafic réel	Mesure active et continue, même sans trafic ; vue utilisateur depuis le réseau ; corrélation possible avec les traces de l'APM (chapitre 18)
Métriques et tableaux de bord	Prometheus, Grafana	Collecte, stockage et visualisation des séries temporelles, alerting	Métriques de disponibilité et de performance des services, prêtes à être collectées
Mesure de l'expérience réelle	Outils RUM	Diversité des utilisateurs réels	Détection anticipée, parcours critiques, applications internes
Tests automatisés en intégration continue	Suites de tests fonctionnels	Vérification avant mise en production, sur des environnements de test	Vérification permanente **après** mise en production, dans les conditions réelles d'exploitation

La sonde s'insère dans ce paysage : ses résultats sont publiés vers vos plateformes existantes, où ils rejoignent vos tableaux de bord, vos règles d'alerte et vos processus d'astreinte. Vos équipes conservent leurs outils et leurs habitudes.

2.9 L'apport de la supervision applicative aux équipes

2.9.1 Pour les équipes informatiques

Les équipes d'exploitation sont alertées sur les incidents qui affectent réellement les utilisateurs, avec les éléments nécessaires à leur qualification. Elles peuvent prioriser leurs interventions selon l'impact réel, plutôt que selon le nombre d'alarmes techniques. Les équipes de développement et de qualité vérifient après chaque mise en production que les parcours critiques fonctionnent toujours, et repèrent les dégradations de performance introduites par une nouvelle version. Les responsables d'infrastructure disposent d'indicateurs objectifs pour piloter les engagements de service, internes comme externes, et pour dialoguer avec leurs prestataires.

2.9.2 Pour les métiers

Les responsables métier obtiennent une mesure de la disponibilité exprimée dans leurs termes : « ce mois-ci, le parcours de commande a réussi dans 99,7 % des exécutions évaluables », plutôt que « le serveur web a été disponible ». Ce taux est une proportion d'exécutions, non une proportion du temps : il s'accompagne de la couverture de la mesure, c'est-à-dire de la part des exécutions attendues qui ont effectivement produit un résultat évaluable, car les périodes sans mesure et les changements de fréquence peuvent dissocier les deux grandeurs. Une disponibilité exprimée en temps, par exemple dans un engagement de niveau de service, suppose une méthode de calcul explicitement convenue avec les parties prenantes (chapitre 17, § 17.7.3). Les indicateurs d'activité (commandes, transactions, inscriptions) surveillés par la sonde les alertent sur des anomalies qu'eux seuls savent interpréter, et que la seule supervision technique ne révèle pas. La sonde fournit enfin un langage commun aux équipes informatiques et métier pour décrire la qualité de service.

2.9.3 Bonnes conditions de réussite

L'expérience montre que la valeur d'un projet de supervision applicative dépend moins de l'outil que de quelques choix d'organisation :

Condition	Recommandation
Choisir les bons parcours	Commencer par trois à cinq parcours critiques pour l'activité (connexion, recherche, commande, paiement en environnement de test), plutôt que de chercher à tout couvrir
Définir les attendus avec les métiers	Formaliser pour chaque parcours le résultat attendu et le délai acceptable
Disposer de comptes et de données de test	Prévoir des comptes dédiés, des articles de test et des environnements où les actions sont sans conséquence (chapitre 4, § 4.6)
Organiser la réaction aux alertes	Désigner qui reçoit les alertes, à quelle heure, et quelle est la conduite à tenir
Maintenir les scénarios	Associer la mise à jour des scénarios au cycle de mise en production des applications (chapitre 16)

La suite de la documentation décrit comment la sonde met en œuvre ces principes : son architecture (chapitre 3), puis son installation et son utilisation.

Chapitre 3 — Architecture de la solution

Ce chapitre décrit l'organisation de la sonde, ses composants, les échanges qu'elle entretient avec votre système d'information et le déroulement d'une exécution. Il s'adresse aux architectes, aux administrateurs et aux responsables sécurité qui doivent évaluer l'intégration de la sonde dans leur environnement. Les paramètres réseau précis (ports, sens des flux) sont détaillés au chapitre 7 et récapitulés en annexe (chapitre 21).

3.1 Principes d'architecture

L'architecture de la sonde repose sur six principes. Ils expliquent la plupart des choix décrits dans la suite de la documentation.

Principe	Signification	Conséquence pour vous
Autonomie	La sonde planifie, exécute, évalue et conserve elle-même ses contrôles	Elle continue de superviser même si vos plateformes de supervision, votre accès Internet ou les services de l'éditeur sont indisponibles
Source de vérité locale	La configuration et l'historique des résultats sont stockés sur la sonde	Les plateformes externes reçoivent une copie des résultats ; la référence reste consultable localement
Connexions sortantes uniquement vers l'extérieur	Toute communication avec un service extérieur à votre réseau est initiée par la sonde	Aucune ouverture de port depuis Internet n'est nécessaire
Isolation de l'exécution	Les scénarios navigateur et les plugins s'exécutent dans un environnement séparé, sans accès à la configuration ni aux secrets non déclarés	Un script défectueux ou malveillant ne peut ni compromettre la sonde ni lire l'ensemble des identifiants
Standards ouverts	API REST documentée (OpenAPI), OpenTelemetry/OTLP, format Prometheus, codes retour Nagios	Intégration à vos outils sans connecteur propriétaire
Sobriété	Peu de composants, écritures disque maîtrisées, ressources réservées au navigateur	Fonctionnement fiable sur un ordinateur monocarte, 24 heures sur 24

3.2 Vue d'ensemble

La sonde est installée au point d'observation choisi : dans le réseau de votre organisation, à proximité des utilisateurs ou des applications dont vous souhaitez vérifier le service, ou à l'extérieur, sur un site distant, chez un client, un partenaire ou un sous-traitant, ou dans une autre région géographique (§ 3.2.1). Le schéma ci-dessous présente le cas d'une sonde installée dans le réseau de l'entreprise ; les mêmes catégories d'échanges s'appliquent aux autres emplacements. Elle échange avec quatre catégories d'interlocuteurs :

- les **ressources supervisées** : sites web, API, bases de données, serveurs, équipements, sur lesquels elle exécute ses contrôles ;
- les **administrateurs**, qui la configurent et consultent ses résultats depuis un navigateur ou par l'API ;
- les **plateformes de supervision et d'observabilité** (Prometheus, Grafana, Datadog, Dynatrace, Nagios, Centreon…), qui reçoivent ou collectent ses résultats ;

- les **services d'infrastructure** indispensables : DNS, heure (NTP) et, si vous les activez, dépôt de mises à jour de l'éditeur et proxy sortant.

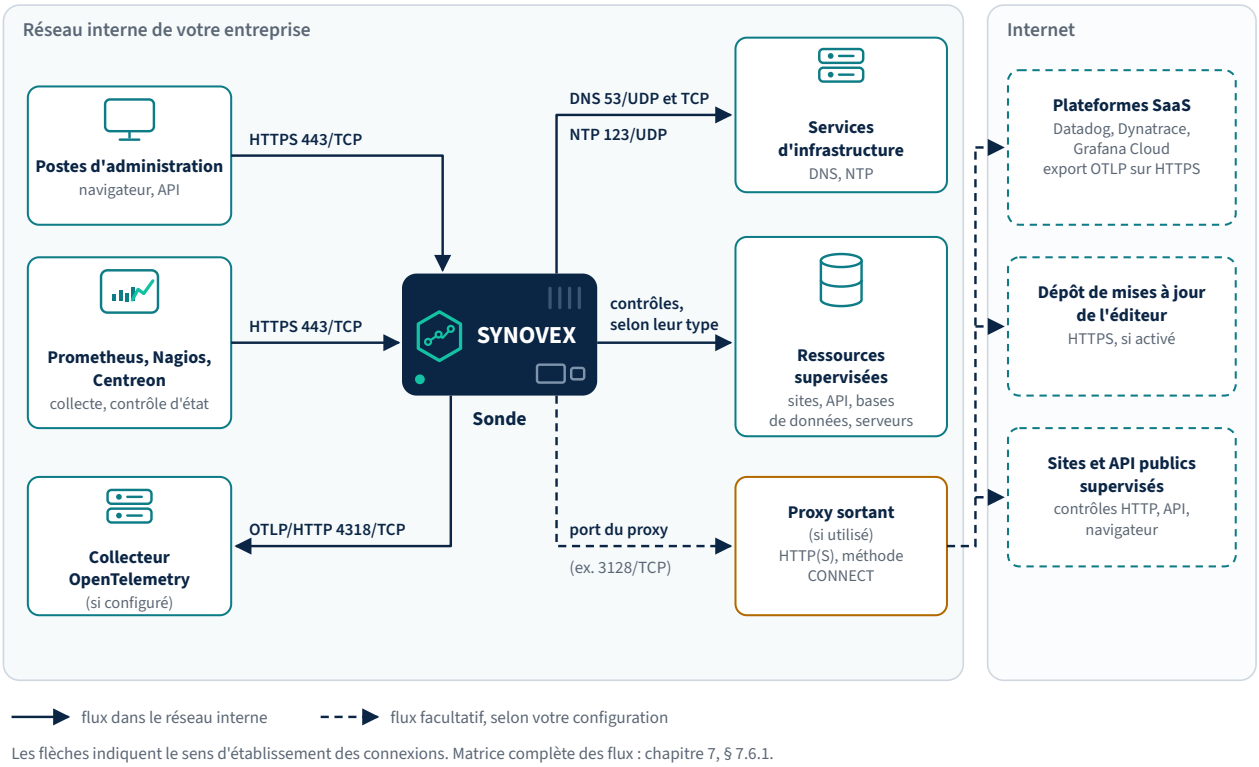


Figure 3-1 — Schéma d'implantation type : sonde dans le réseau interne, plateformes de supervision, accès Internet sortant via proxy

Les flèches indiquent le sens d'établissement des connexions. Les flux en pointillés sont facultatifs et dépendent de votre configuration. La sonde n'envoie aucune donnée de télémétrie à l'éditeur : ses seuls échanges avec lui sont la vérification et le téléchargement des mises à jour, si vous les activez.

3.2.1 Où placer la sonde ?

L'emplacement de la sonde détermine ce qu'elle mesure. Une sonde voit le service tel qu'il est perçu depuis son point d'observation : à travers les mêmes résolveurs DNS, pare-feu, proxys, opérateurs et liens que les utilisateurs ou les applications de ce réseau. Elle peut être installée dans tout environnement qui dispose des accès réseau nécessaires aux contrôles configurés, à l'intérieur comme à l'extérieur de votre organisation.

Emplacement	Ce que la sonde mesure	Usage typique
Réseau des utilisateurs d'un site	Le service tel que le perçoivent les collaborateurs de ce site, y compris les liens intersites et le proxy	Applications métier internes, services SaaS utilisés par l'entreprise
Centre de données, à proximité des serveurs	Le service sans les effets du réseau étendu	Distinguer un problème applicatif d'un problème de réseau
Zone d'accès externe ou réseau isolé vers Internet	Le service tel qu'il est vu depuis l'extérieur, en passant par l'accès Internet	Sites et API publics de l'entreprise
Site distant ou agence : filiale, entrepôt, magasin	Le service tel que le perçoivent les utilisateurs de ce site, à travers son réseau local, son accès opérateur et ses liaisons vers vos centres de données	Vérifier la disponibilité et les performances des applications centrales depuis chaque site

Emplacement	Ce que la sonde mesure	Usage typique
Réseau d'un client, avec son accord	Le service tel que le perçoivent les utilisateurs du client, à travers leur réseau, leur proxy et leur pare-feu	Vérifier que les services proposés par votre organisation sont réellement accessibles et fonctionnels depuis l'environnement du client
Réseau d'un partenaire ou d'un sous-traitant, avec son accord	Les échanges et les services tels qu'ils sont accessibles depuis cette organisation	Superviser un portail partenaire, une interconnexion ou des échanges entre organisations
Différentes régions géographiques	Le service tel qu'il est perçu dans chaque pays ou région : opérateur, liens internationaux, résolution DNS locale, éventuel réseau de diffusion de contenu (CDN)	Mesurer la latence et les temps de réponse par localisation, repérer une dégradation propre à une région ou à un opérateur

Plusieurs sondes placées à des endroits différents fournissent des **indices de localisation** d'un défaut. Si le même contrôle échoue depuis un site et réussit depuis le centre de données, l'écart oriente l'analyse vers ce qui distingue les deux points d'observation, souvent le chemin réseau du site. De même, un contrôle qui réussit depuis Paris mais échoue ou ralentit depuis Hô Chi Minh-Ville oriente l'analyse vers le chemin international, l'opérateur ou les services régionaux. Ces indices ne suffisent pas à conclure : un composant réseau commun peut affecter tous les sites, et une réponse qui dépend de la région, de l'identité du compte de test, du répartiteur de charge ou de la résolution DNS peut n'affecter qu'un seul point d'observation alors que la cause est applicative. Avant d'attribuer l'origine à l'application ou au réseau, comparez les chemins, les cibles effectivement jointes, les résolutions DNS, les politiques appliquées (proxy, filtrage) et les résultats détaillés des sondes. Chaque sonde rend compte de ce qu'elle observe depuis son emplacement ; elle n'identifie pas elle-même la cause d'une dégradation réseau, et chaque sonde fonctionne indépendamment des autres. Chaque sonde se voit attribuer un **nom de site** lors de sa mise en service (chapitre 6), repris dans tous ses résultats exportés : c'est lui qui permet de comparer les sondes entre elles dans vos tableaux de bord et vos alertes.

Les prérequis d'une sonde installée hors de vos locaux dépendent des services à superviser et de leur accessibilité depuis ce réseau. Prévoyez en particulier les points suivants :

- **accord du propriétaire du réseau** : installez-la uniquement avec l'autorisation écrite de l'organisation qui exploite le réseau, et convenez avec elle des cibles contrôlées, des flux ouverts et de la personne qui intervient physiquement sur l'appareil ;
- **transmission des résultats** : privilégiez l'envoi par OTLP (chapitre 18), qui n'utilise que des connexions sortantes, vers une plateforme joignable depuis ce site. La collecte par Prometheus ou par le plugin Nagios suppose au contraire que votre plateforme puisse joindre la sonde ;
- **administration** : l'interface et l'API ne doivent pas être exposées sur Internet (§ 3.7). Administrez la sonde depuis le réseau où elle est installée ou au travers d'un moyen d'accès sécurisé convenu avec l'exploitant de ce réseau. La sonde ne requiert aucune liaison permanente avec votre organisation pour exécuter ses contrôles ;
- **coupure de liaison** : la sonde continue d'exécuter ses contrôles et de conserver leur historique localement lorsque la liaison vers vos plateformes est interrompue. Les résultats exportés pendant la coupure ne sont pas renvoyés automatiquement ; l'historique local reste consultable (chapitre 18, § 18.12) ;
- **heure et réglementation locales** : vérifiez qu'un serveur NTP est joignable depuis le site et que l'installation respecte les règles locales applicables à l'équipement et aux données collectées.

3.3 Composants de la sonde

La sonde réunit un matériel, un système d'exploitation et deux moteurs logiciels.

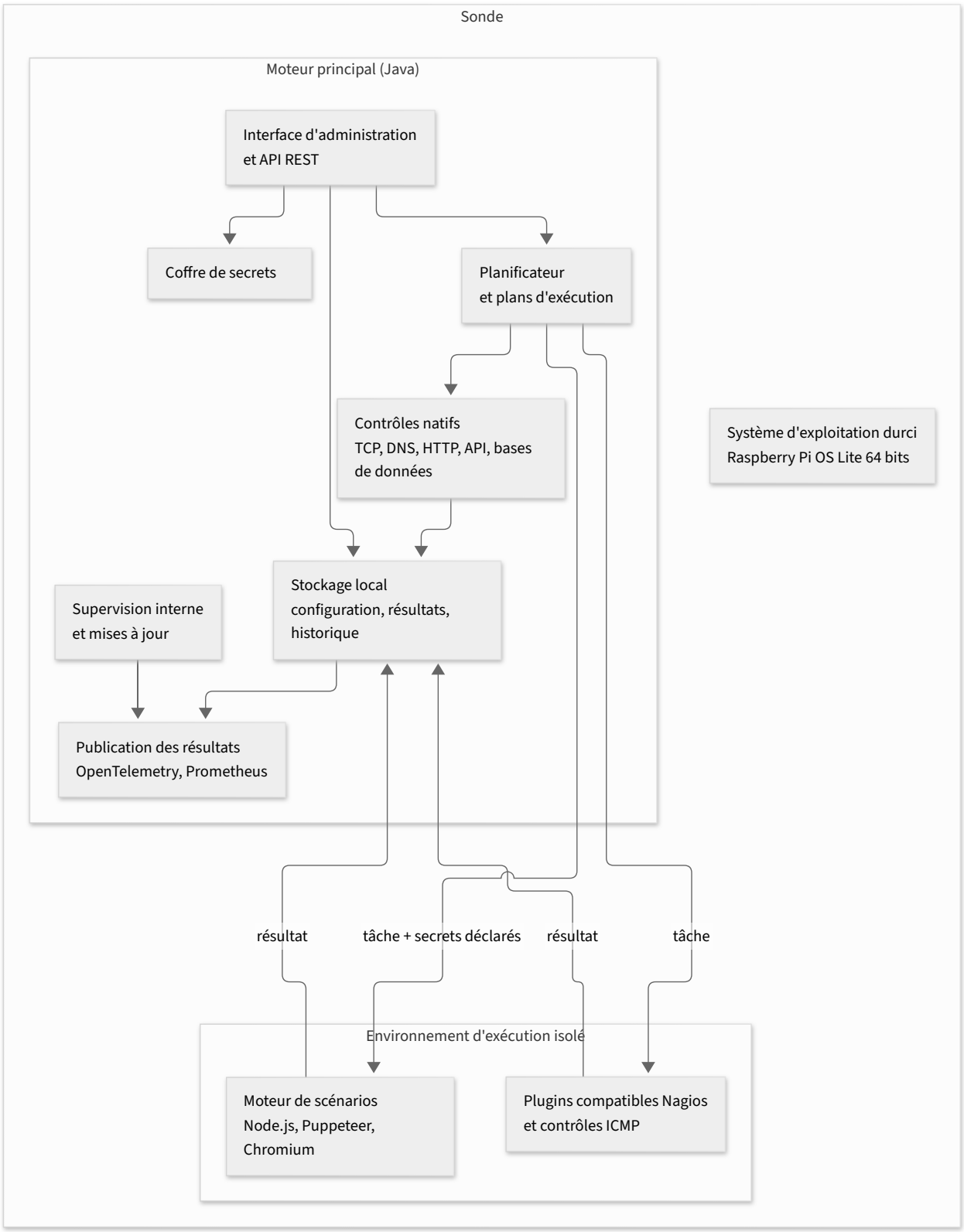


Schéma 3-1 — Composants de la sonde

3.3.1 Matériel et système

La sonde fonctionne sur un **Raspberry Pi 5**, ordinateur monocarte à processeur ARM 64 bits, avec une capacité mémoire adaptée à l'édition (chapitre 4). Le système d'exploitation est **Raspberry Pi OS Lite 64 bits**, distribution dérivée de Debian maintenue par le fabricant de la carte, installé sans interface graphique et durci : services inutiles retirés, accès distant SSH désactivé par défaut, comptes et droits limités au strict nécessaire. Le système est configuré pour limiter les écritures sur la carte microSD, dont la durée de vie dépend du volume écrit (chapitre 20).

La carte microSD porte deux **emplacements système** (A et B), dont un seul est actif. Le système y est monté en **lecture seule** : il n'est pas modifié en fonctionnement normal, de sorte qu'une coupure électrique n'interrompt aucune écriture du système. Cette disposition ne protège pas contre une défaillance du support lui-même (chapitre 20, § 20.3). Une mise à jour s'installe sur l'emplacement inactif, et la sonde revient automatiquement à l'emplacement précédent si la nouvelle version ne fonctionne pas correctement (§ 20.4.5). Les données de la sonde sont stockées sur une partition séparée de la microSD. Un **SSD NVMe** facultatif, monté sur la carte M.2 HAT+ officielle, peut accueillir les données les plus volumineuses ; la sonde continue de démarrer sur la microSD (chapitre 20, § 20.3.4).

3.3.2 Moteur principal

Le **moteur principal** est le cœur de la sonde. Développé en Java, il fonctionne en permanence et assure les fonctions suivantes.

Fonction	Description
Interface d'administration et API REST	Serveur HTTPS qui publie l'interface web et l'API documentée ; authentification des utilisateurs et des jetons, contrôle des droits, journal d'audit (chapitre 9). Le moteur d'authentification repose sur des fournisseurs interchangeables : la version actuelle utilise des comptes locaux, et l'ajout d'OpenID Connect est prévu dans une version ultérieure (chapitre 9, § 9.4)
Planification	Calcul des échéances de chaque contrôle et de chaque plan d'exécution, répartition dans le temps pour éviter les pics, limitation du nombre d'exécutions simultanées (chapitre 12)
Contrôles natifs	Exécution directe des contrôles `tcp`, `dns`, `http`, `api` et `database`. Les connexions aux bases de données utilisent les pilotes standard JDBC de chaque éditeur de base (chapitre 11)
Évaluation	Application des critères de réussite (assertions, seuils), des nouvelles tentatives et des règles de calcul de l'état (chapitre 8)
Stockage local	Enregistrement de la configuration, des résultats, des étapes et des agrégats dans une base SQLite locale ; purge selon la durée de rétention
Coffre de secrets	Conservation chiffrée des mots de passe et jetons utilisés par les contrôles ; transmission limitée aux secrets déclarés par chaque contrôle
Publication	Production de métriques, traces et journaux au format OpenTelemetry ; export OTLP vers les destinations configurées ; endpoint Prometheus (chapitre 18)
Supervision interne	Surveillance de l'état de la sonde elle-même : mémoire, température, alimentation, espace disque, retards de planification, échecs d'export
Mises à jour	Vérification, téléchargement, contrôle de signature et installation des images système sur l'emplacement inactif, sans interrompre les contrôles ; canaux `security` (automatique par défaut) et `feature` (manuel par défaut) ; activation automatique dans une fenêtre de maintenance définie par le client, après préavis et drainage des exécutions, ou, par exception, à la demande d'un administrateur ou dès que possible pour un correctif critique si le client l'a choisi ; validation après redémarrage et retour arrière automatique (chapitre 20, § 20.4)

3.3.3 Environnement d'exécution isolé

Certains contrôles exécutent du code qui ne provient pas de l'éditeur : scripts de scénarios écrits par vos équipes, plugins compatibles Nagios. Ils s'exécutent dans un **environnement isolé**, distinct du moteur principal :

- un compte système dédié, sans accès en lecture à la base de configuration ni au coffre de secrets ;
- des limites de mémoire, de processeur et de nombre de processus, qui empêchent un scénario défaillant d'épuiser les ressources de la sonde ;
- un processus neuf pour chaque exécution, arrêté de force si le délai maximal est dépassé ;
- la réception, pour chaque exécution, des seuls secrets que le contrôle a explicitement déclarés.

Le **moteur de scénarios** fait partie de cet environnement. Il repose sur **Node.js**, environnement d'exécution JavaScript, sur **Puppeteer**, bibliothèque de pilotage du navigateur, et sur **Chromium** en mode headless. Pour chaque exécution d'un scénario, il démarre un navigateur neuf avec un profil vierge conservé en mémoire, exécute les étapes, mesure leur durée, réalise une capture d'écran en cas d'échec, puis ferme le navigateur. Aucune session, aucun cookie ni aucune donnée de navigation n'est conservé d'une exécution à l'autre. La version de Chromium est fournie et validée par l'éditeur avec la version de Puppeteer correspondante (chapitre 13).

Les **plugins compatibles Nagios** et les contrôles **ICMP** (ping), qui reposent sur des utilitaires système, s'exécutent dans ce même environnement isolé, quelle que soit l'édition. Les plugins sont lancés directement, sans interpréteur de commandes, ce qui empêche toute injection de commande par leurs arguments.

> **Information** — Cette séparation explique pourquoi la création de scénarios scriptés et l'ajout de plugins personnalisés sont réservés aux administrateurs : ces opérations reviennent à exécuter du code sur le réseau où la sonde est installée. Le chapitre 9 détaille les rôles.

3.3.4 Répartition des types de contrôles

Type de contrôle	Composant d'exécution	Édition minimale
`icmp` (ping)	Environnement isolé, utilitaire système	Essential
`tcp`, `dns`, `http`	Moteur principal	Essential
`nagios`, plugins fournis	Environnement isolé	Essential
`nagios`, plugins personnalisés	Environnement isolé	Professional
`api`	Moteur principal	Professional
`database`	Moteur principal (JDBC)	Professional
`browser` (scénarios déclaratifs et scriptés)	Moteur de scénarios	Enterprise

Chaque famille d'exécution dispose de sa propre limite de simultanéité. Un parcours navigateur long ne retarde donc pas les contrôles réseau, qui s'exécutent en parallèle dans leur propre file. Par défaut, un seul parcours navigateur s'exécute à la fois, car Chromium est le composant le plus consommateur de mémoire. Les plans d'exécution permettent d'enchaîner plusieurs scénarios sans les faire se chevaucher (chapitre 12).

3.4 Stockage et conservation des données

La sonde conserve localement toutes les informations nécessaires à son fonctionnement et à son exploitation.

Donnée	Contenu	Conservation par défaut
Configuration	Contrôles, plans d'exécution, destinations d'export, paramètres, comptes, jetons (sous forme d'empreinte)	Permanente, avec sauvegarde automatique quotidienne (trois générations)
Secrets	Mots de passe et jetons utilisés par les contrôles, chiffrés	Permanente, jusqu'à suppression
Exécutions détaillées	Résultat, état, durée, étapes, erreur, mesures de chaque exécution	14 jours
Agrégats horaires	Nombre d'exécutions, d'échecs, durées représentatives, par contrôle et par heure	400 jours
Artefacts	Captures d'écran, journaux de console du navigateur, sorties de plugins, produits en cas d'échec	7 jours, dans la limite de 500 Mo
Journal d'audit	Opérations d'administration	365 jours
Journaux système	Messages techniques des composants	En mémoire, effacés au redémarrage (par défaut)

Ces données sont stockées sur la partition de données de la microSD. Si un SSD NVMe est activé, les exécutions détaillées, les agrégats, les artefacts, la file d'attente d'export et les journaux persistants y sont placés ; la configuration, les secrets et le certificat d'édition restent sur la microSD (chapitre 20, § 20.3.4). Ces durées sont configurables dans les limites de l'espace disponible (chapitre 9). Le stockage est conçu pour limiter les effets d'une coupure électrique : une coupure peut faire perdre les derniers résultats enregistrés ; la configuration, enregistrée avec un niveau de durabilité supérieur, et la cohérence des bases sont préservées tant que le support de stockage fonctionne correctement, et la base est vérifiée à chaque démarrage. Une panne de la carte elle-même relève de la sauvegarde de la configuration. Le chapitre 20 détaille les mesures de protection de la carte microSD et les procédures de sauvegarde et de restauration.

3.5 Cycle de vie d'une exécution

Toutes les exécutions suivent le même cycle, quel que soit le type de contrôle. L'exemple ci-dessous décrit celui d'un scénario navigateur, le plus complet.

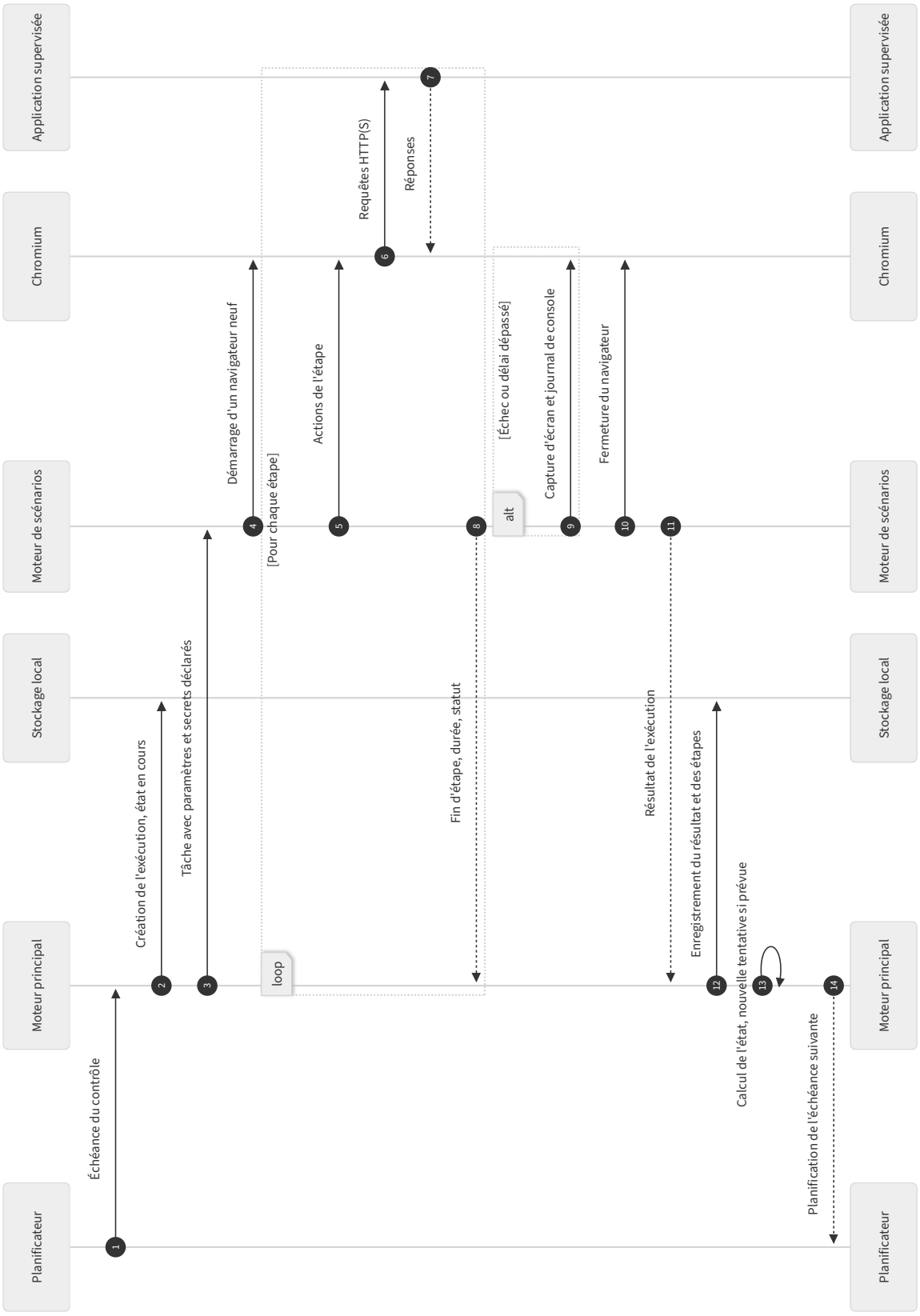


Schéma 3-2 — Cycle de vie d'une exécution (scénario navigateur)

Schéma présenté en orientation paysage : tournez le document d'un quart de tour pour le lire.

Les étapes communes à tous les contrôles sont les suivantes :

1. **Déclenchement.** Le planificateur déclenche le contrôle à son échéance, ou un utilisateur le déclenche manuellement depuis l'interface ou l'API.
2. **Préparation.** Le moteur principal crée l'exécution, résout les paramètres et récupère les secrets déclarés par le contrôle.
3. **Exécution.** Le contrôle est exécuté par le composant approprié, dans la limite du délai maximal (*timeout*) configuré.
4. **Évaluation.** Le résultat est comparé aux critères du contrôle : assertions, seuils de durée, codes retour.
5. **Confirmation.** En cas d'échec, le contrôle peut être relancé selon les nouvelles tentatives configurées avant que l'état `CRITICAL` soit retenu.
6. **Enregistrement.** Le résultat, les étapes, les mesures et les éventuels artefacts sont enregistrés localement.
7. **Publication.** Les métriques, la trace de l'exécution et les journaux associés sont mis à disposition des destinations configurées.
8. **Replanification.** L'échéance suivante est calculée.

Le chapitre 8 détaille les notions d'état, de délai, de nouvelle tentative et de seuil utilisées à chaque étape.

3.6 Publication et interopérabilité des résultats

La sonde ne se contente pas de conserver ses résultats : elle les publie selon des formats standard, pour que vous les exploitiez dans vos outils. Les mêmes résultats alimentent trois canaux, qui peuvent être utilisés simultanément.

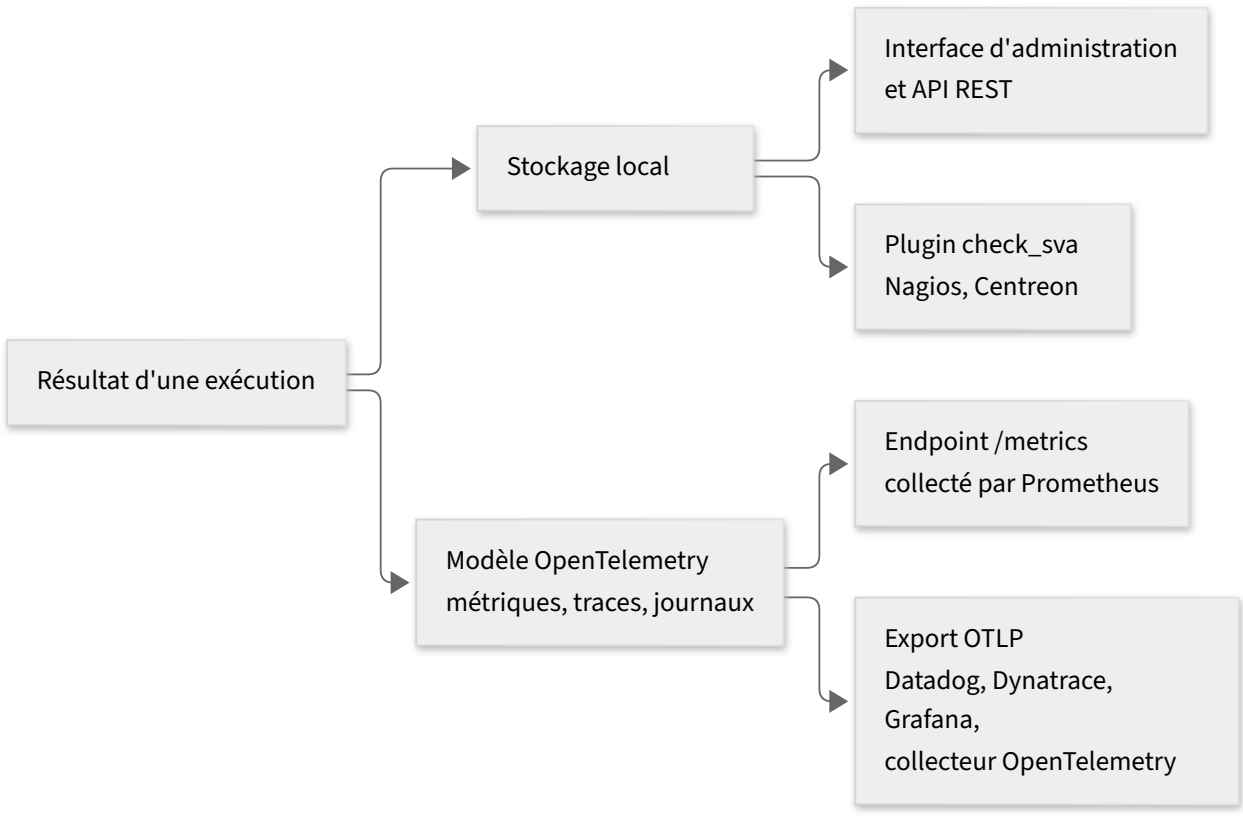


Schéma 3-3 — Publication des résultats : stockage local, API et exports

Canal	Principe	Plateformes
Export OTLP (envoi)	La sonde envoie métriques, traces et journaux selon le protocole OpenTelemetry (principalement OTLP/HTTP)	Collecteur OpenTelemetry, Datadog (via son agent ou en direct), Dynatrace, Grafana Cloud et toute plateforme compatible OTLP
Endpoint Prometheus (collecte)	La sonde expose ses métriques au format Prometheus ; votre serveur Prometheus vient les collecter	Prometheus, et par son intermédiaire Grafana
Plugin de contrôle d'état (interrogation)	Un plugin installé sur votre serveur Nagios ou Centreon interroge l'API de la sonde et renvoie l'état d'un contrôle selon les conventions Nagios	Nagios, Centreon et outils compatibles

OpenTelemetry est un standard ouvert, soutenu par l'ensemble des acteurs de l'observabilité, qui définit comment décrire et transmettre trois types de données : les **métriques** (valeurs numériques mesurées dans le temps, par exemple la durée d'un parcours), les **traces** (déroulement détaillé d'une opération, ici une exécution et ses étapes) et les **journaux** (événements horodatés, par exemple l'échec d'une étape avec son message). Le choix de ce standard évite de développer un connecteur spécifique à chaque plateforme.

> ! **Attention** — Le transport standardisé garantit que vos plateformes **reçoivent** les données. Il ne crée pas automatiquement les tableaux de bord ni les règles d'alerte. Ces éléments se configurent dans chaque plateforme ; le chapitre 18 décrit la démarche et les modèles fournis.

La publication est découplée de l'exécution : si une destination d'export est indisponible, la sonde poursuit ses contrôles, conserve les résultats localement et réessaie l'envoi dans des limites définies. Les conséquences d'une indisponibilité prolongée sur chaque type de données sont décrites au chapitre 18.

3.7 Administration de la sonde

La sonde s'administre **localement**, depuis le réseau où elle est installée, sans dépendre d'un service extérieur.

Moyen	Usage	Chapitre
Assistant de mise en service	Configuration initiale : compte administrateur, réseau, heure, proxy, certificat, destinations d'export	6
Fichier de provisionnement	Configuration initiale sans intervention interactive, préparée sur la carte microSD avant le premier démarrage ; adapté aux déploiements en nombre	6
Interface d'administration web	Consultation des états et des résultats, gestion des contrôles, des plans d'exécution, des utilisateurs, des secrets, des destinations d'export, des paramètres	9
API REST	Automatisation de toutes les opérations de l'interface : intégration à vos outils de déploiement, gestion en masse, synchronisation avec vos référentiels	9 et 21
Accès SSH	Maintenance avancée, désactivé par défaut et activable par un administrateur	7 et 20

L'interface d'administration utilise exclusivement l'API REST : tout ce que vous faites dans l'interface peut donc être automatisé. Tous les échanges sont chiffrés en HTTPS. Les accès sont authentifiés par compte nominatif ou par jeton d'API, et soumis à quatre rôles : administrateur (`admin`), opérateur (`operator`), lecteur (`viewer`) et collecte de métriques (`metrics`) (chapitre 9). Chaque opération d'administration est consignée dans un journal d'audit.

→ **Évolution prévue** — L'authentification par OpenID Connect, pour se connecter avec les comptes de l'annuaire de l'entreprise, est prévue dans une version ultérieure. Elle s'ajoutera comme un nouveau fournisseur d'authentification, avec une correspondance entre les groupes de l'annuaire et les rôles de la sonde, sans modification des rôles ni de l'API.

! **Avertissement** — L'interface d'administration et l'API ne doivent jamais être exposées directement sur Internet. La sonde est conçue pour être administrée depuis le réseau où elle est installée ; l'accès à distance passe par un moyen d'accès sécurisé à ce réseau, celui de votre organisation ou celui convenu avec l'exploitant du réseau d'accueil.

3.8 Modes de fonctionnement

La sonde connaît trois modes de fonctionnement successifs, auxquels s'ajoutent un mode restreint lié au certificat d'édition et un état temporaire de maintenance, propre à l'activation des mises à jour.

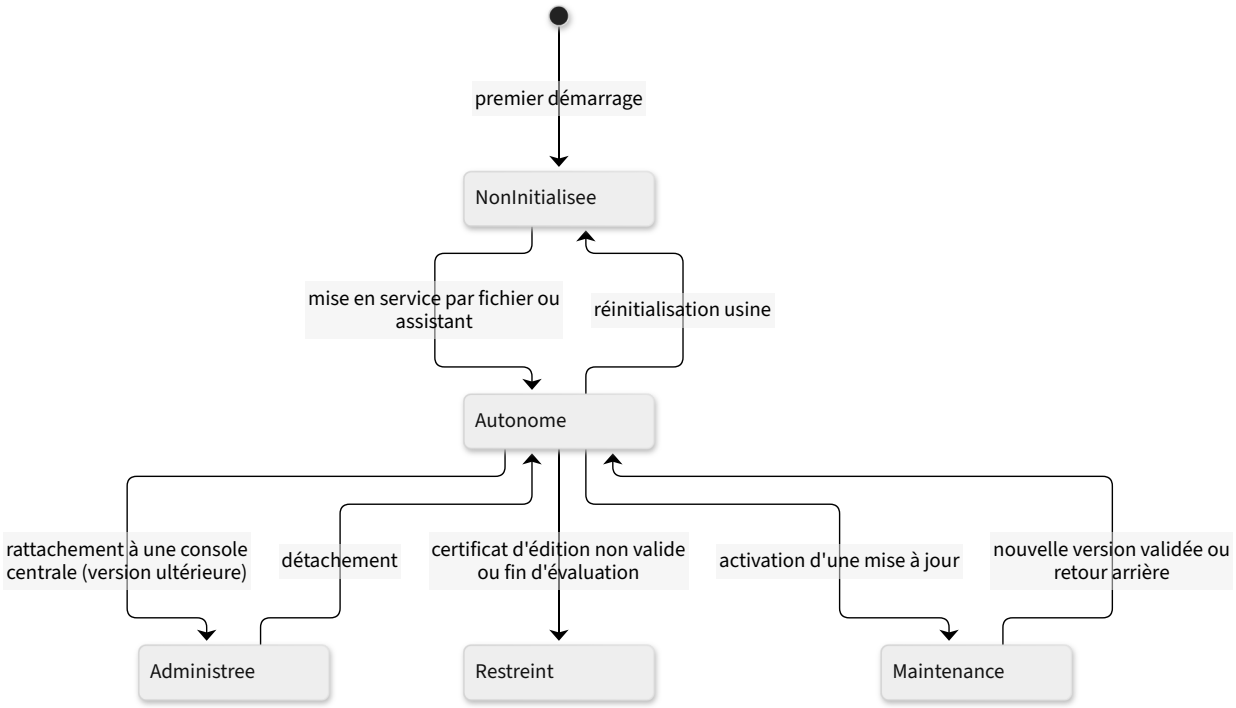


Schéma 3-4 — Modes de fonctionnement de la sonde

Mode	Description
Non initialisée	État à la sortie de l'emballage, ou après une réinitialisation. La sonde n'exécute aucun contrôle et attend sa mise en service, par fichier de provisionnement ou par l'assistant protégé par un code d'installation (chapitre 6).
Autonome	Mode nominal. La configuration est gérée localement par l'interface ou l'API ; les résultats sont conservés localement et publiés vers les destinations configurées.
Administrée	Mode prévu avec la console centrale. La configuration est définie dans la console et appliquée par la sonde, qui continue de fonctionner avec la dernière configuration reçue si la console devient injoignable.

Mode	Description
Maintenance	État temporaire de l'activation d'une mise à jour, dans une fenêtre définie par le client, à la demande d'un administrateur ou, pour un correctif critique avec `criticalPolicy: asap`, dès que possible et hors fenêtre. La sonde **draine** d'abord ses exécutions : aucune nouvelle exécution ne démarre et celles en cours se terminent dans un délai limité. Elle **redémarre** ensuite sur la nouvelle version, qui s'autoteste ; en cas d'échec, elle revient à la version précédente. Cet état est exposé par `/health` (`MAINTENANCE`) et par les métriques de la sonde (chapitre 20, § 20.4).
Restreint	Mode appliqué lorsque le certificat d'édition est absent, invalide ou propre à une autre carte, ou au terme des 30 jours d'évaluation d'une carte de démonstration prêtée. Aucun contrôle n'est exécuté ; la consultation, l'export de la configuration et l'archive de diagnostic restent disponibles. La cause est indiquée dans l'état de santé (§ 6.5).

Une sonde ne supervise pas pendant son propre redémarrage. Lorsque la continuité de la supervision est critique, déployez au moins deux sondes par site, exécutant les mêmes contrôles avec des fenêtres de maintenance qui ne se chevauchent pas, séparées d'un écart supérieur à la durée totale mesurée d'une activation, et fondez vos alertes sur l'ensemble des sondes du site (chapitre 20, § 20.4). Cet espacement réduit le risque que deux sondes du même site soient en maintenance en même temps sans l'exclure : il ne coordonne ni les activations manuelles ni celles d'un correctif critique en mode `asap`, qui ont lieu hors fenêtre. Pour ces dernières, un décalage fixe propre à chaque sonde (`security.criticalStagger`) échelonne les activations, sans vérifier l'état des autres sondes : les sondes étant autonomes, rien ne garantit l'absence de redémarrage simultané (chapitre 20, § 20.4.11 et § 20.4.12). N'activez jamais manuellement deux sondes d'un même site en même temps.

> → **Évolution prévue** — La **console centrale multi-sondes**, hébergée par l'éditeur ou installée dans votre infrastructure, permettra de gérer un parc de sondes depuis un point unique : inventaire, configuration commune, synthèse des états. La sonde s'y connectera par une liaison sortante chiffrée et authentifiée par certificat. Cette console restera facultative : une sonde administrée continue de fonctionner sans elle, et une sonde autonome n'en a jamais besoin. L'adresse de la console hébergée par l'éditeur, `https://fleet-synovex.incitypark.fr`, est réservée dans le plan de nommage ; la sonde ne la contacte pas.

3.9 Sécurité intégrée

La sécurité est prise en compte à tous les niveaux. Le tableau ci-dessous résume les mesures, détaillées au chapitre 7 (réseau et sécurité) et au chapitre 9 (comptes, rôles et secrets).

Domaine	Mesures
Mise en service	Aucun mot de passe par défaut ; assistant protégé par un code d'installation propre à la carte ; certificat HTTPS propre à chaque sonde, généré au premier démarrage
Accès	HTTPS uniquement ; comptes nominatifs ; jetons d'API à portée limitée, révocables ; quatre rôles ; protection contre les tentatives répétées ; SSH désactivé par défaut
Secrets	Chiffrés au repos ; jamais relisibles par l'API ; transmis aux seuls contrôles qui les déclarent ; masqués dans les journaux
Exécution	Environnement isolé pour les scénarios et plugins ; compte système dédié ; limites de ressources ; navigateur neuf à chaque exécution ; pas d'interpréteur de commandes pour les plugins
Réseau	Aucune ouverture entrante depuis Internet ; flux sortants limités et documentés ; prise en charge d'un proxy ; IPv4, IPv6 activable ; authentification 802.1X par certificat (EAP-TLS) sur le port Ethernet (chapitre 7)
Traçabilité	Journal d'audit des opérations d'administration, exportable vers votre outil de gestion des événements de sécurité

Domaine	Mesures
Mises à jour	Images système signées par l'éditeur, installées sur un emplacement A/B avec retour arrière automatique (§ 20.4.5) ; système en lecture seule ; version de Chromium validée ; correctifs de sécurité pendant toute la période de support, sur un canal distinct des versions fonctionnelles et installés automatiquement par défaut (désactivable) ; activation automatique dans les fenêtres choisies par le client, avec préavis et report possible ; exceptions documentées : activation manuelle et correctif critique en mode `asap` (chapitre 20)

3.10 Comparaison avec l'architecture historique

Les clients de la génération précédente de la solution reconnaîtront certains principes, mais l'architecture a profondément évolué. Le tableau suivant facilite la transition.

Aspect	Génération précédente	SYNOVEX
Dépendance à un serveur	Serveur central indispensable pour stocker, agréger et afficher les résultats	Aucune ; sonde autonome, console centrale facultative (évolution prévue)
Liaison avec l'extérieur	Connexion WebSocket permanente vers le serveur central	Connexions sortantes facultatives : exports OTLP, mises à jour
Pilotage du navigateur	CasperJS et PhantomJS, technologies abandonnées	Puppeteer et Chromium headless, maintenus
Contrôles de web services	Projets SoapUI	Contrôles HTTP et API natifs avec assertions
Contrôles de bases de données	Fichiers de règles	Contrôles `database` configurés par l'interface ou l'API, avec les mêmes opérateurs de comparaison
États	`OK`, `WARNING`, `CRITICAL`, `STALE`	Mêmes états, complétés par `UNKNOWN`
Intégration Nagios	Contrôle passif via le serveur central	Plugin actif interrogeant directement la sonde ; export OpenTelemetry et Prometheus
Déploiement des scénarios	Copie manuelle de fichiers	Interface d'administration et API

Les fonctions métier de la génération précédente sont conservées : parcours web, contrôles de web services, contrôles de bases de données, captures d'écran en cas d'échec, historique, rejeu d'un contrôle. Les anciens scénarios CasperJS ne sont pas exécutables par la nouvelle sonde ; le chapitre 13 (§ 13.7) donne la correspondance entre les anciennes pratiques et Puppeteer pour faciliter leur réécriture.

PARTIE II

Installation et mise en service

Chapitre 4 — Prérequis

Chapitre 5 — Installation physique

Chapitre 6 — Premier démarrage et mise en service

Chapitre 7 — Configuration réseau et sécurité

Chapitre 4 — Prérequis

Ce chapitre recense tout ce qui doit être réuni **avant** l'installation de la sonde : matériel, alimentation, carte microSD, raccordement réseau, autorisations de flux, comptes de test, poste d'administration et conditions d'installation. Il se termine par une liste de vérification à compléter avant le premier démarrage (§ 4.9).

> ✓ **Conseil** — Transmettez ce chapitre à vos équipes réseau et sécurité dès la commande de la sonde. Les demandes d'ouverture de flux et la création de comptes de test demandent souvent plusieurs jours ; les anticiper évite de retarder la mise en service.

4.1 Matériel compatible

4.1.1 Carte supportée

La sonde fonctionne **exclusivement sur Raspberry Pi 5**. La carte microSD contient un système préparé pour ce seul modèle ; son fonctionnement sur un autre modèle de Raspberry Pi ou sur une carte d'un autre fabricant n'est ni pris en charge ni garanti.

> ! **Attention** — N'insérez pas la carte microSD de la sonde dans un Raspberry Pi 4, un Raspberry Pi 400, un Compute Module ou une carte Orange Pi. Le système de la sonde contrôle le modèle de carte au tout début de son démarrage, avant tout montage d'une partition en écriture et avant l'extension de la partition de données (§ 6.2). Sur un matériel autre qu'un Raspberry Pi 5, deux cas se présentent. Si le microprogramme (*firmware*) de la carte parvient à charger le système, celui-ci s'arrête sans monter de partition en écriture et affiche un message sur l'écran HDMI éventuellement raccordé. Si le microprogramme d'un autre modèle ne charge pas l'image, aucun composant de la sonde ne s'exécute et aucun message ne peut s'afficher ; cela n'établit pas à lui seul qu'aucune écriture n'a eu lieu, le comportement du microprogramme tiers n'étant pas maîtrisé par la sonde. Dans les deux cas, la sonde ne démarre pas ; ce contrôle vise à préserver la carte pour qu'elle reste utilisable ensuite sur le bon matériel, sans que sa conservation puisse être garantie lorsque le microprogramme d'une autre carte a manipulé le support. La prise en charge d'autres cartes est une évolution envisagée, non disponible actuellement.

4.1.2 Capacité mémoire selon l'édition

Le Raspberry Pi 5 existe en plusieurs capacités de mémoire vive. La capacité requise dépend de l'édition, principalement parce que le navigateur Chromium utilisé par les scénarios de l'édition Enterprise est gourmand en mémoire.

Édition	Mémoire minimale	Mémoire recommandée	Justification
Essential	2 Go	4 Go	Contrôles réseau, HTTP et plugins : faible consommation. La marge de 4 Go facilite une évolution ultérieure vers Professional
Professional	4 Go	4 Go ou plus	Pilotes de bases de données et suites de requêtes API
Enterprise	**8 Go**	8 Go	Chromium consomme, à titre indicatif, plusieurs centaines de mégaoctets par parcours en cours d'exécution

La capacité de mémoire est indiquée sur l'emballage du Raspberry Pi 5 ; après la mise en service, elle est également affichée dans l'état de santé de la sonde (chapitre 6), et la réponse de `GET /api/v1/system` l'indique dans le champ `resources.memoryTotalBytes`, en octets. Pour l'offre « boîtier complet », le Raspberry Pi 5 livré a la capacité de mémoire requise par l'édition.

> **Information** — Le changement d'édition se fait par échange de la carte microSD contre une carte de l'édition supérieure, avec transfert de la configuration (chapitre 20, § 20.6). Si vous envisagez de passer à l'édition Enterprise, prévoyez dès le départ un Raspberry Pi 5 de 8 Go.

4.1.3 Composition des offres

Élément	Offre « microSD préconfigurée »	Offre « boîtier complet »
Carte microSD préconfigurée	Fournie	Fournie, déjà insérée
Raspberry Pi 5	À fournir par vous (capacité selon § 4.1.2)	Fourni
Boîtier	À fournir par vous (§ 4.4)	Fourni, assemblé
Refroidissement	À fournir par vous (§ 4.4)	Fourni, monté
Alimentation USB-C	À fournir par vous (§ 4.2)	Selon la composition de l'offre, indiquée sur la liste de colisage de votre commande ; à défaut, à fournir par vous (§ 4.2)
Câble Ethernet	À fournir par vous	Selon la composition de l'offre, indiquée sur la liste de colisage de votre commande ; à défaut, à fournir par vous
Carte M.2 HAT+ officielle et SSD NVMe (facultatifs, § 4.3.3)	À fournir par vous, si vous choisissez cette option	Selon votre commande ; à défaut, à fournir par vous si vous choisissez cette option
Encart d'identification de la carte (édition, numéro de série, code d'installation)	Fourni	Fourni
Étiquette d'identification du boîtier (numéro de série, adresse MAC, nom d'hôte par défaut, code d'installation, édition et capacité mémoire)	Non	Fournie
Accessoires de fixation (montage en baie, fixation murale)	Non	Selon votre commande

La liste de colisage exacte et les références des composants sont fournies avec votre commande.

> **Information** — Ne présumez pas qu'un accessoire est fourni s'il ne figure pas sur la liste de colisage de votre commande. Avant l'installation, vérifiez en particulier l'alimentation (§ 4.2), le câble Ethernet et, le cas échéant, la carte M.2 HAT+ et le SSD. Seuls les éléments qui définissent chaque offre sont toujours présents : la carte microSD et son encart pour l'offre « microSD préconfigurée » ; le Raspberry Pi 5 en boîtier, son refroidissement et la carte insérée pour l'offre « boîtier complet ».

4.1.4 Écran, clavier et souris

La sonde n'a besoin ni d'écran, ni de clavier, ni de souris : elle s'administre par le réseau. Lors du premier démarrage, l'empreinte du certificat doit toutefois être vérifiée par un canal indépendant du réseau : un écran raccordé au port micro-HDMI de la carte, ou la lecture du fichier `setup-code.txt` sur la microSD (§ 6.4.4). L'écran est aussi utile pour

lire le code d'installation et l'empreinte du certificat de la sonde, et suivre le démarrage (chapitre 6), ou lors d'un dépannage. Le Raspberry Pi 5 dispose de ports **micro-HDMI** : prévoyez un câble ou un adaptateur micro-HDMI vers HDMI si nécessaire.

4.1.5 Compatibilité avec d'autres ordinateurs monocartes

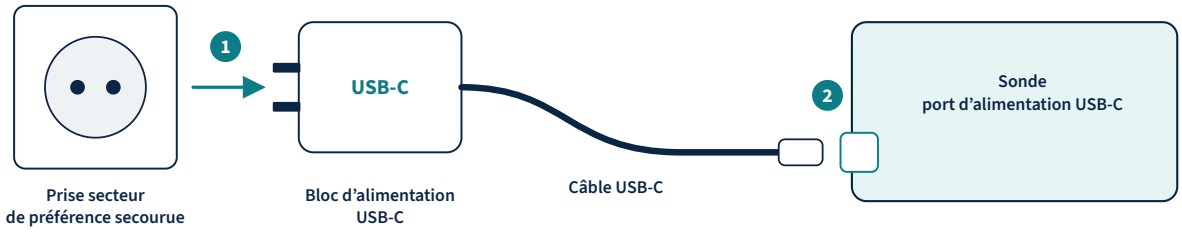
SYNOVEX est conçu et documenté pour Raspberry Pi 5. Son architecture logicielle peut permettre une utilisation sur d'autres ordinateurs monocartes, notamment certains modèles Orange Pi. Toutefois, ces plateformes n'ont pas fait l'objet d'une validation officielle et leur fonctionnement n'est pas garanti.

La carte microSD livrée contient une image système propre au Raspberry Pi 5 : chargeur d'amorçage, noyau et mécanismes de démarrage et de mise à jour de cette carte. Elle ne peut pas être utilisée telle quelle sur une carte Orange Pi ou sur un autre ordinateur monocarte, ni passer d'un modèle de carte à l'autre (§ 4.1.1). Les procédures, les valeurs et les exigences de cette documentation concernent le seul Raspberry Pi 5.

4.2 Alimentation électrique

4.2.1 Caractéristiques requises

La qualité de l'alimentation est la première cause d'instabilité des ordinateurs monocartes : une alimentation insuffisante provoque des redémarrages inopinés, des erreurs d'écriture et, à terme, la corruption de la carte microSD.



Ordre de branchement

1. Brancher le bloc d'alimentation sur la prise secteur.
2. Brancher le connecteur USB-C sur la sonde, en dernier.

Raspberry Pi 5 : exigences (chapitre 4, § 4.2.1)	
Recommandée	Alimentation officielle Raspberry Pi 27 W USB-C (5,1 V, 5 A)
Alternative	Alimentation USB-C 5 V / 3 A de qualité : courant limité sur les ports USB Non admise avec un SSD NVMe
Avec un SSD NVMe	Alimentation officielle de 27 W obligatoire
À proscrire	Chargeurs de téléphone génériques, ports USB d'ordinateur ou d'écran, câbles USB-C longs ou de faible section, multiprises non protégées

Représentation schématique, non à l'échelle

Figure 4-1 — Raccordement de l'alimentation du Raspberry Pi 5 : prise, bloc d'alimentation USB-C, câble et sonde

Caractéristique	Valeur
Alimentation recommandée	**Alimentation officielle Raspberry Pi 27 W USB-C** (5,1 V, 5 A)
Connecteur	USB-C

Caractéristique	Valeur
Alternative	Alimentation USB-C 5 V / 3 A de qualité : la carte fonctionne, mais limite le courant disponible sur ses ports USB (comportement documenté par Raspberry Pi). **Non admise avec un SSD NVMe** (§ 4.3.3)
À proscrire	Chargeurs de téléphone génériques, ports USB d'ordinateur ou d'écran, câbles USB-C longs ou de faible section, multiprises non protégées

> **Attention** — Une alimentation sous-dimensionnée peut sembler fonctionner pendant des jours avant de provoquer des incidents difficiles à diagnostiquer. La sonde détecte les épisodes de sous-tension et les signale dans sa supervision interne (chapitre 17), mais seule une alimentation conforme les évite. Pour l'édition Enterprise, l'alimentation officielle de 27 W est fortement recommandée. Elle est **obligatoire** lorsque la sonde est équipée d'un SSD NVMe (§ 4.3.3).

4.2.2 Continuité électrique

La sonde est conçue pour qu'une coupure électrique n'affecte pas sa configuration, tant que le support de stockage fonctionne correctement (chapitre 3, § 3.4). Les coupures répétées augmentent néanmoins le risque d'usure et de corruption de la carte microSD.

> **Information** — Une coupure brutale peut faire perdre les toutes dernières écritures (résultats d'exécution, événements encore en attente d'envoi). Les bases sont conçues pour rester cohérentes, seules les dernières transactions pouvant manquer, à condition que la carte respecte les écritures synchrones. Une panne du support lui-même (carte défaillante) relève de la sauvegarde de configuration (chapitre 20).

Situation	Recommandation
Installation en salle informatique ou en baie	Raccorder l'alimentation à une prise ondulée (alimentation sans interruption, ASI)
Installation dans un local sans ondulation	Accepter le risque, ou prévoir un petit onduleur ; programmer des sauvegardes de configuration (chapitre 20)
Coupures programmées (travaux électriques)	Arrêter proprement la sonde avant la coupure (chapitre 20)

4.3 Carte microSD

4.3.1 Carte fournie

La carte microSD fournie contient le système d'exploitation, le moteur principal, le moteur de scénarios, Chromium et l'ensemble des composants nécessaires. Elle est prête à l'emploi : aucune installation de logiciel n'est nécessaire.

Caractéristique	Valeur
Capacité	32 Go minimum
Classe	A2 / U3
Type	« High Endurance », conçu pour des écritures continues, recommandé
Contenu	Image système de la sonde ; sa version est indiquée dans les notes de version et, après la mise en service, par `GET /api/v1/system` (champ `version`)

> **Avertissement** — Ne reformatez pas la carte et ne modifiez pas ses partitions. La seule opération prévue sur la carte avant le premier démarrage est le dépôt facultatif d'un fichier de provisionnement sur sa partition `SVA-BOOT` (chapitre 6). Avec le mécanisme de mise à jour A/B, d'autres partitions de la carte peuvent apparaître sur votre poste : ne les modifiez pas. Toute autre modification peut rendre la sonde inutilisable.

4.3.2 Conservation et manipulation

Conservez la carte dans son étui jusqu'à l'installation. Évitez de toucher ses contacts dorés, de la plier, de l'exposer à l'humidité, à la chaleur ou à l'électricité statique. Notez son numéro de série ou la référence de commande avant l'insertion si votre gestion de parc l'exige.

4.3.3 Stockage sur SSD

Les cartes microSD ont une durée de vie limitée par le volume de données écrites. La **carte microSD reste le support principal** de la sonde et **suffit à un usage standard**, quelle que soit l'édition. Un **SSD NVMe** (*Non-Volatile Memory Express*, disque à mémoire flash raccordé par le bus PCI Express) peut la compléter. Il est **facultatif** et surtout recommandé pour l'édition Enterprise, dont les scénarios navigateur produisent davantage d'écritures.

Le SSD sert uniquement de **volume de données** :

Support	Contenu
Carte microSD (toujours)	Partition de démarrage, deux emplacements système (mécanisme de mise à jour, chapitre 20), certificat d'édition, configuration et secrets
SSD NVMe (s'il est activé)	Historique des exécutions, agrégats, artefacts (captures d'écran), file d'attente d'export, journaux persistants

La sonde démarre toujours sur la microSD : l'ordre de démarrage du Raspberry Pi 5 n'est pas modifié. Le droit d'utilisation reste attaché à la carte microSD. Si le SSD devient absent ou défaillant, les contrôles continuent de s'exécuter, l'historique local est conservé sur la microSD en mode réduit et l'état de santé `storage` passe en `CRITICAL` (chapitre 20, § 20.3.4).

Si vous retenez cette option, réunissez le matériel suivant :

Élément	Exigence
Carte d'extension	**Carte M.2 HAT+ officielle de Raspberry Pi**. Elle fonctionne en PCIe 2.0 x1 (environ 500 Mo/s). Le mode PCIe Gen 3, non certifié par Raspberry Pi, n'est pas utilisé par la sonde
SSD	SSD NVMe au format M.2 **2230 ou 2242**, figurant dans la liste des modèles pris en charge, publiée dans les notes de version
Alimentation	**Alimentation officielle Raspberry Pi 27 W USB-C**, obligatoire avec un SSD (§ 4.2.1)
Refroidissement	Compatible avec le refroidisseur actif officiel (*Active Cooler*), qui doit être **installé en premier**, avant la carte M.2 HAT+ (chapitre 5)
Boîtier	Le boîtier officiel Raspberry Pi 5 n'est utilisable **que sans son couvercle ni son ventilateur**. Avec un autre boîtier, vérifiez auprès de son fabricant qu'il accepte une carte M.2 HAT+

Le montage du matériel est décrit au chapitre 5 (§ 5.4.6) et l'activation du SSD par la sonde au chapitre 20 (§ 20.3.4).

> **Attention** — L'activation **formate entièrement** le SSD. N'utilisez pas un SSD qui contient des données à conserver.

4.3.4 Carte de rechange

Pour les sondes dont la disponibilité est critique, il est recommandé de disposer d'une carte microSD préconfigurée de rechange et d'une sauvegarde récente de la configuration. La remise en service consiste alors à remplacer la carte, à effectuer la mise en service puis à restaurer la configuration (chapitre 20).

4.4 Refroidissement et boîtier

Le processeur du Raspberry Pi 5 réduit automatiquement sa fréquence lorsqu'il chauffe. Ce bridage thermique ralentit les contrôles, notamment les parcours navigateur, et fausse les mesures de durée.

Édition	Refroidissement
Essential, Professional	Refroidisseur actif officiel ou boîtier ventilé **recommandé**
Enterprise	Refroidisseur actif officiel ou boîtier ventilé **obligatoire**

Pour l'offre microSD préconfigurée, choisissez un boîtier qui :

- laisse accessibles le lecteur de carte microSD, le port Ethernet, le connecteur USB-C d'alimentation et, si possible, le bouton d'alimentation ;
- assure une circulation d'air suffisante ou intègre un ventilateur ;
- protège la carte des contacts accidentels et de la poussière ;
- se fixe de manière stable dans votre environnement : étagère, baie, support mural.

La sonde surveille la température de son processeur et signale tout bridage thermique (chapitre 17).

> **Information** — Si la sonde reçoit un SSD NVMe (§ 4.3.3), le refroidisseur actif officiel reste utilisable : il s'installe en premier, sous la carte M.2 HAT+. Le boîtier officiel Raspberry Pi 5 ne peut alors être utilisé que sans son couvercle ni son ventilateur.

4.5 Réseau

4.5.1 Raccordement

La sonde se raccorde **exclusivement en Ethernet filaire**. Le Raspberry Pi 5 dispose d'un port Ethernet Gigabit RJ45.

Élément	Exigence
Câble	Câble Ethernet catégorie 5e ou supérieure, longueur adaptée
Prise réseau	Port de commutateur actif, dans le réseau ou le VLAN choisi pour la sonde
Débit	100 Mbit/s suffisent ; 1 Gbit/s pris en charge
Alimentation par Ethernet (PoE)	Non prise en charge : la carte d'extension officielle PoE+ (HAT PoE+) n'est pas prévue pour la sonde. Alimentez la sonde par son port USB-C (§ 4.2)
Contrôle d'accès au réseau (802.1X)	Pris en charge par certificat (méthode EAP-TLS) (§ 7.2.6) ; exception par adresse MAC (MAB) possible en alternative. Prérequis ci-dessous

Si le port de raccordement est protégé par le contrôle d'accès **802.1X**, norme selon laquelle un équipement doit s'authentifier auprès du commutateur avant d'obtenir l'accès au réseau, préparez avec votre équipe réseau :

- un **serveur RADIUS** (*Remote Authentication Dial-In User Service*, serveur qui valide les authentifications transmises par le commutateur) acceptant la méthode **EAP-TLS**, c'est-à-dire l'authentification par certificat ;
- un **certificat client** pour la sonde, émis par votre infrastructure de gestion de clés (IGC), avec l'identité que la sonde présentera au serveur RADIUS. Il est recommandé de faire générer la clé et la demande de certificat (CSR) par la sonde elle-même, après une première mise en service sur un port non contrôlé : la clé privée ne quitte alors jamais l'appareil. Si la sonde doit s'authentifier dès son premier démarrage, prévoyez à la place un fichier PKCS#12 (clé et certificat) et son mot de passe, déposés sur la microSD avec le fichier de provisionnement (chapitre 6) ;
- le certificat de l'**autorité de certification (AC)** qui a émis le certificat du serveur RADIUS, au format PEM ;
- à défaut, ou pour la seule mise en service, une **exception par adresse MAC** (MAB, *MAC Authentication Bypass*) sur le port de la sonde ;
- de préférence, la **pile officielle de l'horloge temps réel** (RTC, *Real-Time Clock*) du Raspberry Pi 5, installée sur la carte. La vérification des certificats EAP-TLS dépend de l'heure, alors que les serveurs NTP ne sont joignables qu'après l'authentification : la pile conserve l'heure lorsque la sonde est hors tension. Sans elle, l'heure de départ est la dernière heure enregistrée par la sonde ou, à défaut, la date de construction de l'image (§ 7.2.6).

La procédure complète figure au chapitre 7 (§ 7.2.6).

> **Attention** — La sonde ne prend en charge ni l'étiquetage VLAN (IEEE 802.1Q) sur la prise de la sonde, ni la méthode PEAP (authentification 802.1X par identifiant et mot de passe). Raccordez la sonde à un port d'accès configuré dans le VLAN choisi.

> **Évolution prévue** — La connexion Wi-Fi n'est pas prise en charge. Le Wi-Fi et le Bluetooth intégrés au Raspberry Pi 5 sont désactivés dans le système de la sonde, par une surcouche de l'arbre des périphériques (*device tree overlay*).

4.5.2 Choix de l'emplacement réseau

L'emplacement de la sonde détermine ce qu'elle mesure (chapitre 3, § 3.2.1). Avant l'installation, déterminez :

1. le **réseau ou VLAN** de raccordement, en fonction des services à superviser et du point de vue souhaité (utilisateurs d'un site, centre de données, accès Internet, site distant ou à l'étranger, réseau d'un client ou d'un partenaire) ;
2. le **mode d'adressage** : adresse attribuée par DHCP (par défaut) ou adresse IP fixe. Une adresse fixe, ou une réservation DHCP, est recommandée pour qu'administrateurs et plateformes de supervision joignent toujours la sonde à la même adresse. L'IPv4 est obligatoire ; l'IPv6 est facultatif et désactivé par défaut. Activez-le si des ressources à superviser ne sont joignables qu'en IPv6, ou si vous souhaitez contrôler leur accès en IPv6 (chapitre 7, § 7.2.5) ;
3. les **serveurs DNS** que la sonde doit utiliser. Ils déterminent les noms qu'elle sait résoudre ; utilisez les mêmes que les utilisateurs dont vous voulez reproduire le point de vue ;
4. les **serveurs NTP**. Une heure exacte est indispensable à la cohérence des résultats, des certificats et des exports ;
5. le **proxy sortant** éventuel, nécessaire si la sonde doit joindre des sites externes, une plateforme SaaS ou le dépôt de mises à jour et que votre réseau impose un proxy.

Si vous choisissez une adresse fixe, préparez les informations suivantes :

Paramètre	Exemple fictif	Votre valeur
Adresse IP et masque	`192.0.2.50/24`	
Passerelle	`192.0.2.1`	
Adresse IPv6 et passerelle IPv6, si l'IPv6 est activé en mode statique	`2001:db8:10::50/64`, `2001:db8:10::1`	
Serveurs DNS	`192.0.2.10`, `192.0.2.11`	
Domaine de recherche DNS	`example.internal`	
Serveurs NTP	`ntp1.example.internal`	
Proxy sortant et exclusions	`http://proxy.example.internal:3128`, sauf `.example.internal`	
Nom d'hôte souhaité	`sonde-paris-01`	
Nom de site	`paris-dc1`	

4.5.3 Résumé des flux réseau

Les flux nécessaires sont détaillés au chapitre 7 et récapitulés dans le tableau des flux réseau du chapitre 21, à transmettre à vos équipes réseau et sécurité. En synthèse :

Sens	Flux	Obligatoire
Entrant vers la sonde	HTTPS (TCP 443) depuis les postes d'administration et, le cas échéant, depuis Prometheus, Nagios ou Centreon	Oui, depuis le réseau d'administration
Entrant vers la sonde	HTTP (TCP 80), redirection vers HTTPS uniquement	Non : facultatif, désactivable
Entrant vers la sonde	mDNS (UDP 5353), découverte du nom `.local` sur le réseau local	Non : désactivable
Entrant vers la sonde	SSH (TCP 22) depuis les postes d'administration	Non : désactivé par défaut
Sortant	DHCP, DNS, NTP	Oui (DHCP si utilisé)
Entrant et sortant	ICMPv6 de découverte des voisins, sur le réseau local de la sonde	Si l'IPv6 est activé
Couche 2, entre la sonde et le commutateur	EAPOL (authentification 802.1X), sans flux IP	Si le port est protégé par 802.1X
Sortant	Vers chaque ressource supervisée, selon les contrôles : ICMP, ports TCP, HTTP(S), ports des bases de données	Oui
Sortant	Vers les destinations d'export : collecteur OpenTelemetry, agent Datadog, plateformes SaaS	Selon votre configuration
Sortant	Vers le service de distribution des mises à jour de l'éditeur, en HTTPS : `https://updates-synovex.incitypark.fr` (adresse par défaut, inscrite dans le profil éditeur), ou vers votre miroir interne. Sans ce service, les images signées s'importent hors ligne, après leur transmission par le canal convenu dans votre contrat (chapitre 20, § 20.4.4 ; annexe L, § 21.12)	Si mises à jour en ligne

Lorsque l'IPv6 est activé, ces flux s'entendent en IPv4 et en IPv6. Aucun flux entrant depuis Internet n'est nécessaire.

> ✓ **Conseil** — Placez la sonde dans un réseau d'où elle peut atteindre les ressources supervisées, mais limitez ses flux entrants aux seuls postes et serveurs qui en ont besoin. La sonde manipule des identifiants de comptes de test : traitez-la comme un équipement d'infrastructure sensible.

4.6 Prérequis de sécurité et de gouvernance

La sonde agit dans votre système d'information comme un utilisateur automatisé. Son déploiement doit être validé et préparé comme celui de tout compte de service.

4.6.1 Accord des équipes concernées

Avant le déploiement, obtenez l'accord :

- de l'**équipe sécurité**, sur l'installation d'un équipement dans le réseau, les flux, le stockage de secrets et l'usage de comptes de test ;
- des **responsables des applications supervisées**, qui doivent savoir que des transactions automatisées seront exécutées régulièrement, pour ne pas les confondre avec une attaque ou du trafic anormal ;
- de l'**équipe réseau**, pour le raccordement, l'adressage et les ouvertures de flux.

> ! **Attention** — Les contrôles répétés, en particulier les tentatives de connexion, peuvent déclencher des protections : blocage de compte après échecs, limitation de débit, détection d'intrusion, pare-feu applicatif. Faites déclarer l'adresse de la sonde auprès des équipes concernées et prévoyez si nécessaire des exceptions.

4.6.2 Comptes et données de test

Chaque contrôle qui s'authentifie doit utiliser un **compte de test dédié**, jamais un compte personnel ni un compte d'administration.

Exigence	Raison
Compte nominatif « supervision », distinct pour chaque application	Traçabilité ; un compte compromis n'ouvre pas l'accès aux autres applications
Droits minimaux : lecture seule lorsque c'est possible	Limiter l'impact d'une erreur de scénario ou d'une fuite de secret
Mot de passe non soumis à expiration automatique, ou procédure de rotation planifiée	Éviter les fausses alertes à l'expiration du mot de passe
Pas d'authentification multifacteur interactive, ou mécanisme adapté aux comptes de service validé par votre équipe sécurité	Un scénario automatisé ne peut pas saisir un code reçu par téléphone
Données de test identifiables : articles, clients, commandes marqués comme fictifs	Éviter que les actions des scénarios faussent l'activité réelle ou les statistiques
Pour les bases de données : compte en **lecture seule** limité aux tables nécessaires	Une requête de contrôle ne doit jamais pouvoir modifier des données

> **Avertissement** — Un scénario qui ajoute un produit au panier, passe une commande ou effectue un paiement agit réellement sur l'application. N'exécutez ces scénarios en production qu'avec des articles et des moyens de paiement de test prévus à cet effet, et avec l'accord des équipes métier. Le chapitre 15 présente les précautions applicables à chaque exemple.

4.6.3 Secrets

Recensez les secrets dont la sonde aura besoin : mots de passe des comptes de test, jetons d'API des applications supervisées, mots de passe des comptes de base de données, clés d'API des plateformes d'observabilité. Ils seront saisis dans le coffre de secrets de la sonde (chapitre 9) et ne doivent jamais figurer en clair dans la définition d'un contrôle, un script ou un fichier de provisionnement. Prévoyez qui les détient, comment ils sont transmis à l'administrateur de la sonde et à quel rythme ils sont renouvelés.

4.6.4 Certificats

La sonde génère au premier démarrage un certificat autosigné pour son interface HTTPS. Si votre politique impose des certificats émis par votre autorité de certification interne, préparez la demande de certificat (nom d'hôte, éventuellement adresse IP) ; la procédure de remplacement figure au chapitre 7. Si les applications supervisées utilisent des certificats émis par une autorité interne, préparez le certificat de cette autorité pour l'importer dans la sonde, faute de quoi les contrôles HTTPS échoueront avec une erreur de certificat.

4.6.5 Édition et droit d'utilisation

L'édition acquise est inscrite sur la carte microSD lors de sa préparation (certificat d'édition, § 6.5). Elle est vérifiée localement par la sonde, sans connexion à un serveur de l'éditeur. **Aucune clé de licence n'est à préparer ni à saisir** : vérifiez simplement que l'édition indiquée sur l'encart ou l'étiquette correspond à votre commande.

4.7 Poste d'administration

L'administration s'effectue depuis un poste de travail raccordé à un réseau autorisé à joindre la sonde en HTTPS.

Élément	Exigence
Navigateur	Chrome, Edge, Firefox ou Safari, dans l'une de leurs deux dernières versions majeures, ou Firefox ESR
Accès réseau	Accès HTTPS (TCP 443) à l'adresse de la sonde
Lecteur de carte	Lecteur de carte microSD, ou adaptateur SD, si vous utilisez le fichier de provisionnement, si vous devez lire le code d'installation sur la carte, ou, sans écran, pour lire l'empreinte du certificat dans `setup-code.txt` (chapitre 6, § 6.4.4)
Outils facultatifs	Client en ligne de commande pour l'API (par exemple `curl`), client SSH pour la maintenance avancée
Développement de scénarios (Enterprise)	Node.js et Puppeteer installés sur le poste pour tester les scénarios avant déploiement, avec le lanceur local fourni avec les exemples (chapitre 15) ; éditeur de code

4.8 Conditions d'installation physique

Critère	Recommandation
Local	Local technique, baie ou bureau fermé, à accès contrôlé : la carte microSD contient des secrets chiffrés et la configuration de la sonde

Critère	Recommandation
Température ambiante	Plage de température de fonctionnement indiquée par Raspberry Pi Ltd dans la fiche produit du Raspberry Pi 5 ; éviter la proximité de sources de chaleur et l'exposition directe au soleil
Humidité, poussière	Environnement sec et propre, sans condensation
Ventilation	Laisser plusieurs centimètres libres autour des ouïes du boîtier
Fixation	Surface stable, plane, non conductrice, ou fixation prévue par le boîtier ; câbles sans tension
Accès	Accès possible au lecteur de carte microSD, au bouton d'alimentation et aux voyants pour la maintenance
Alimentation	Prise à proximité, de préférence ondulée
Étiquetage	Étiqueter la sonde avec son nom d'hôte, son adresse IP et le contact de l'équipe responsable

> ! **Avertissement** — La protection physique de la sonde fait partie de sa sécurité. Une personne disposant d'un accès physique peut retirer la carte microSD. Les secrets y sont chiffrés, mais la meilleure protection reste un emplacement à accès restreint (chapitre 7).

4.9 Liste de vérification avant démarrage

Complétez cette liste avant de procéder à l'installation physique (chapitre 5). Une version imprimable figure au chapitre 21.

N°	Vérification	Responsable	Fait
1	Édition acquise connue et conforme à la commande (encart ou étiquette)	Chef de projet	[ ]
2	Raspberry Pi 5 de capacité mémoire conforme à l'édition (§ 4.1.2), ou boîtier complet reçu	Technicien	[ ]
3	Alimentation USB-C conforme (§ 4.2), prise de préférence ondulée	Technicien	[ ]
4	Refroidissement et boîtier conformes (§ 4.4)	Technicien	[ ]
5	Carte microSD préconfigurée en bon état, version notée	Technicien	[ ]
6	Emplacement physique choisi et conforme (§ 4.8)	Technicien	[ ]
7	Port réseau actif dans le VLAN choisi ; câble Ethernet disponible	Équipe réseau	[ ]
8	Mode d'adressage décidé : DHCP avec réservation ou IP fixe ; paramètres réseau réunis (§ 4.5.2)	Équipe réseau	[ ]
9	Serveurs DNS et NTP accessibles depuis le réseau de la sonde	Équipe réseau	[ ]
10	Proxy sortant identifié, le cas échéant	Équipe réseau	[ ]
11	Flux entrants HTTPS autorisés depuis les postes d'administration	Équipe sécurité	[ ]
12	Flux sortants vers les ressources supervisées autorisés	Équipe sécurité	[ ]
13	Flux vers les destinations d'export et le dépôt de mises à jour autorisés, si utilisés	Équipe sécurité	[ ]
14	Accord des responsables des applications supervisées ; adresse de la sonde déclarée	Chef de projet	[ ]
15	Comptes de test créés, avec droits minimaux (§ 4.6.2)	Responsables applicatifs	[ ]
16	Secrets recensés et circuit de transmission défini (§ 4.6.3)	Équipe sécurité	[ ]
17	Certificat d'autorité interne disponible, si des applications utilisent une AC interne (§ 4.6.4)	Équipe sécurité	[ ]

N°	Vérification	Responsable	Fait
18	Mode de mise en service choisi : fichier de provisionnement ou assistant (chapitre 6)	Administrateur	[ ]
19	Poste d'administration avec navigateur récent et accès au réseau de la sonde (§ 4.7)	Administrateur	[ ]
20	Lecteur de carte microSD disponible, si nécessaire (§ 4.7)	Administrateur	[ ]
21	Premiers contrôles à créer identifiés, avec leurs critères de réussite	Responsables applicatifs	[ ]
22	Destinataires des alertes et plateforme de supervision cible identifiés (chapitre 18)	Exploitation	[ ]
23	Si le port est protégé par 802.1X : serveur RADIUS, certificat client ou exception MAB, certificat de l'AC du serveur RADIUS préparés ; pile officielle de l'horloge temps réel (RTC) recommandée (§ 4.5.1)	Équipe réseau	[ ]
24	Si un SSD NVMe est prévu : carte M.2 HAT+ officielle, SSD pris en charge et alimentation de 27 W réunis (§ 4.3.3)	Technicien	[ ]

Lorsque toutes les lignes sont cochées, vous pouvez procéder à l'installation physique décrite au chapitre 5.

Chapitre 5 — Installation physique

Ce chapitre décrit l'installation matérielle de la sonde, depuis l'ouverture du colis jusqu'au premier allumage. Deux parcours sont proposés, selon l'offre acquise :

- **Offre microSD préconfigurée** : vous fournissez le Raspberry Pi 5 et ses accessoires, puis vous assemblez l'ensemble ;
- **Offre boîtier complet**, prêt à l'emploi : la sonde vous est livrée assemblée ; il suffit de l'installer et de la raccorder.

Les § 5.1 à § 5.3 concernent les deux offres. Le § 5.4 est propre à l'offre microSD préconfigurée, le § 5.5 à l'offre boîtier complet. Les § 5.6 à § 5.10 s'appliquent de nouveau aux deux.

> **Information** — Ce chapitre traite uniquement du matériel et du raccordement. La configuration logicielle (adresse IP, compte administrateur, vérification de l'édition) fait l'objet du chapitre 6. Il est recommandé de lire le chapitre 6 **avant** d'allumer la sonde : selon la méthode de mise en service choisie, un fichier de provisionnement doit parfois être déposé sur la microSD avant le premier démarrage (voir § 6.3).

5.1 Vue d'ensemble de la procédure

Le schéma suivant résume l'enchaînement des opérations.

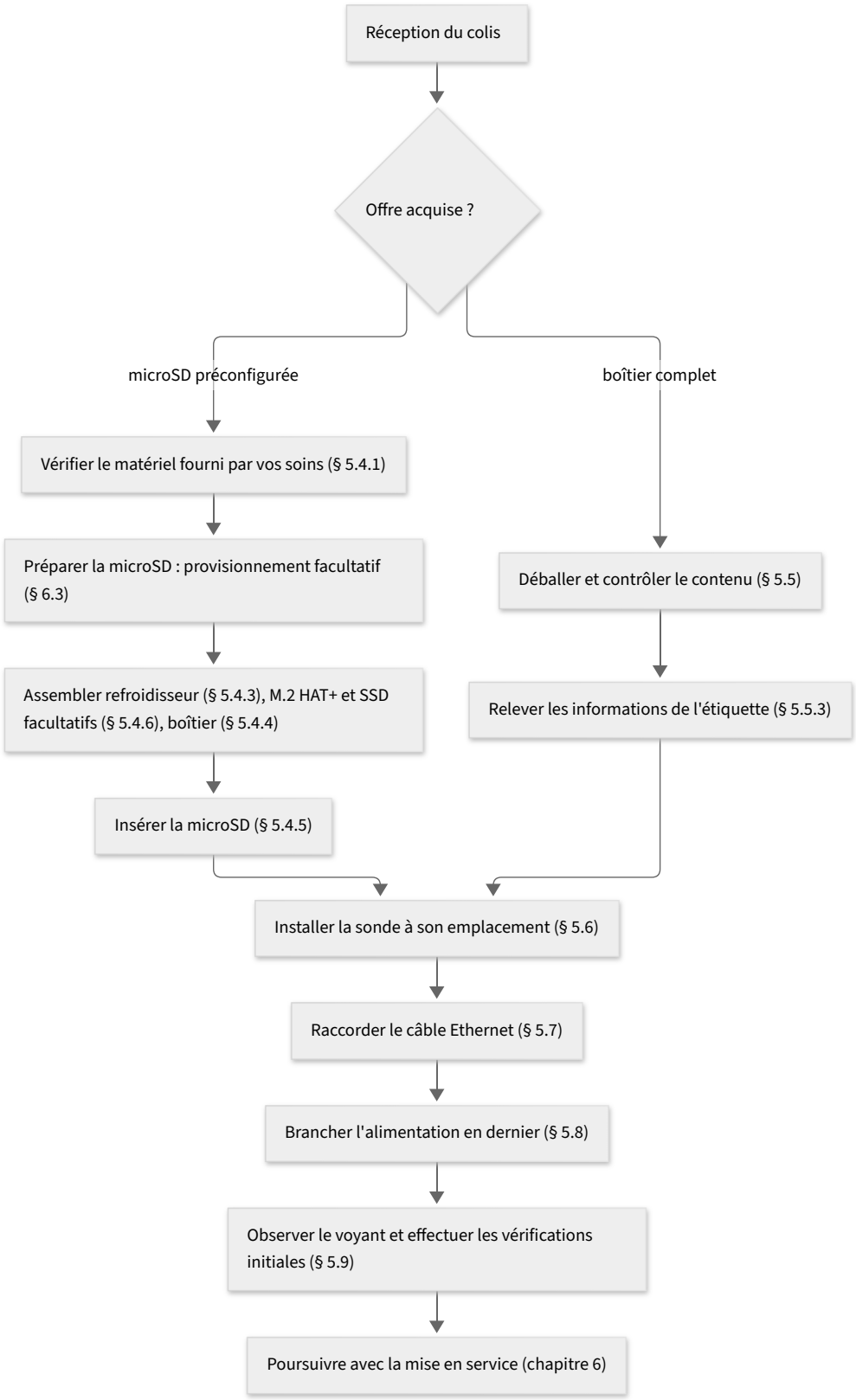


Schéma 5-1 — Enchaînement des opérations d'installation physique

Durée indicative : 10 minutes pour le boîtier complet, 20 à 30 minutes pour la microSD préconfigurée, hors préparation éventuelle du fichier de provisionnement.

5.2 Précautions de manipulation

Le Raspberry Pi 5 est une carte électronique nue. Comme tout équipement de ce type, il est sensible aux décharges électrostatiques, aux contraintes mécaniques et à l'humidité.

> **Avertissement** — Ne manipulez jamais la carte, la microSD ou les accessoires lorsque l'alimentation est branchée. Débranchez toujours le câble d'alimentation USB-C avant toute intervention à l'intérieur du boîtier.

5.2.1 Protection contre les décharges électrostatiques

Une décharge électrostatique (ESD, *ElectroStatic Discharge*) est le transfert brutal de charges électriques accumulées sur le corps humain vers un composant électronique. Elle est souvent imperceptible pour l'opérateur, mais peut endommager durablement un composant.

Appliquez les règles suivantes :

1. Travaillez sur une surface non conductrice et dégagée (table en bois ou tapis antistatique). Évitez la moquette et les vêtements en laine ou en fibres synthétiques.
2. Avant de saisir la carte, touchez une partie métallique reliée à la terre (châssis d'une baie raccordée à la terre, par exemple) ou portez un bracelet antistatique relié à la terre.
3. Tenez la carte **par ses bords**. Ne touchez ni les composants, ni les broches du connecteur GPIO, ni les contacts dorés de la microSD.
4. Conservez la carte dans son sachet antistatique d'origine jusqu'au moment de l'assemblage.

5.2.2 Précautions mécaniques

- N'exercez aucune pression sur les composants lors de la pose du refroidisseur ou de la fermeture du boîtier.
- N'insérez ni ne retirez la microSD en forçant : elle n'entre que dans un seul sens (voir § 5.4.5).
- Ne pliez pas les nappes ou câbles fins (câble du ventilateur notamment) à angle vif.
- Ne posez pas d'objet sur la sonde une fois installée : la circulation d'air doit rester libre.

5.2.3 Précautions liées à l'environnement

- N'installez pas la sonde dans un lieu humide, poussiéreux ou exposé aux projections de liquide.
- Ne l'exposez pas au rayonnement direct du soleil ni à proximité d'une source de chaleur.
- Respectez la plage de température ambiante de fonctionnement indiquée par Raspberry Pi Ltd dans la fiche produit du Raspberry Pi 5 (§ 4.8).
- Laissez le matériel s'acclimater environ une heure s'il a été transporté par temps froid, afin d'éviter toute condensation lors de la mise sous tension.

5.3 Identification du matériel

5.3.1 Le Raspberry Pi 5

La sonde repose exclusivement sur le **Raspberry Pi 5**. Aucun autre modèle n'est pris en charge par l'image fournie : son fonctionnement sur un Raspberry Pi 4 ou sur une autre carte n'est ni pris en charge ni garanti (§ 4.1.1).

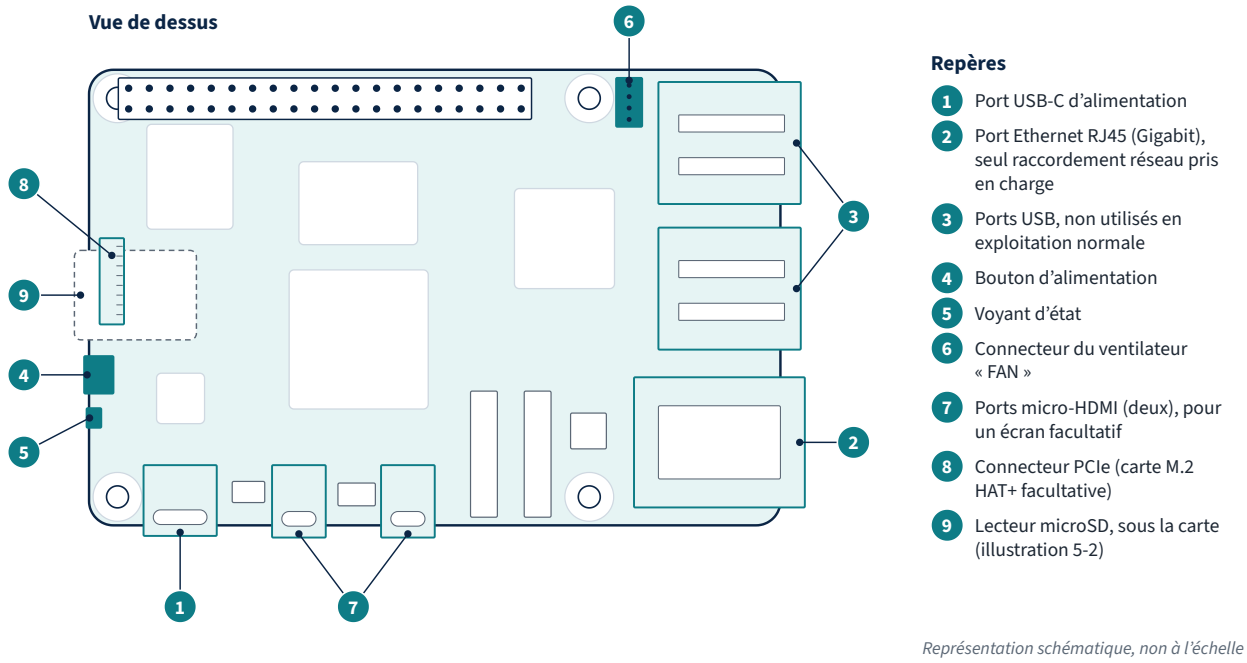


Figure 5-1 — Vue de dessus du Raspberry Pi 5 avec repérage des éléments utiles : port USB-C d'alimentation, port Ethernet, ports USB, bouton d'alimentation, voyant d'état, connecteur du ventilateur, ports micro-HDMI

Le tableau suivant recense les éléments de la carte utilisés lors de l'installation.

Élément	Emplacement	Usage pour la sonde
Port USB-C	Grand côté de la carte, à côté des deux ports micro-HDMI	Alimentation électrique
Port Ethernet RJ45 (Gigabit)	Petit côté de la carte, avec les ports USB	Raccordement au réseau — **seul mode de connexion pris en charge**
Lecteur de carte microSD	**Face inférieure** de la carte, sur le petit côté opposé aux ports Ethernet et USB	Accueille la microSD contenant le système
Bouton d'alimentation	Bord de la carte (repère sur l'figure 5-1)	Mise en marche et arrêt. Comportement standard de Raspberry Pi OS sur Raspberry Pi 5 : sonde arrêtée, un appui la démarre ; sonde en fonctionnement, un appui bref déclenche l'arrêt propre et un appui long force la coupure
Voyant d'état	Bord de la carte (repère sur l'figure 5-1)	Indique l'état d'alimentation et l'activité (voir § 5.9.1)
Connecteur du ventilateur (marqué « FAN »)	Face supérieure (repère sur l'figure 5-1)	Raccordement du refroidisseur actif ou du ventilateur du boîtier
Ports micro-HDMI (deux)	Même côté que le port USB-C	Facultatif : affichage du code d'installation et de l'empreinte du certificat sur un écran (voir § 6.4.3)

Élément	Emplacement	Usage pour la sonde
Ports USB	Côté du port Ethernet	Non utilisés en exploitation normale

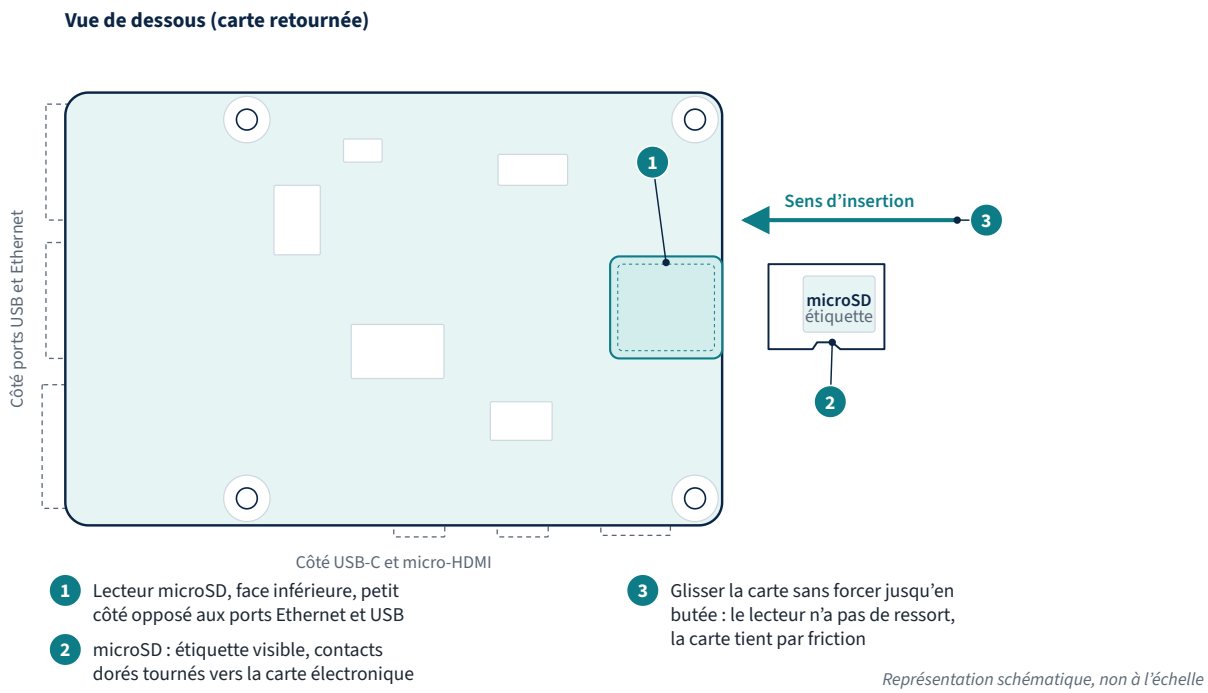


Figure 5-2 — Vue de dessous du Raspberry Pi 5 : emplacement du lecteur microSD, flèche indiquant le sens d'insertion de la carte

5.3.2 La quantité de mémoire

Le Raspberry Pi 5 existe en plusieurs capacités de mémoire vive (RAM). La capacité requise dépend de l'édition de la sonde.

Édition	RAM minimale	RAM recommandée
Essential	2 Go	4 Go
Professional	4 Go	4 Go ou plus
Enterprise	**8 Go**	8 Go

La capacité est indiquée sur l'emballage de la carte ; après la mise en service, elle est également visible dans l'état de santé de la sonde (§ 6.6) : la réponse de `GET /api/v1/system` l'indique dans le champ `resources.memoryTotalBytes`, en octets. Les raisons de ces exigences, liées notamment au navigateur Chromium utilisé par les scénarios de l'édition Enterprise, sont détaillées au chapitre 4.

5.3.3 La microSD préconfigurée

La microSD fournie contient le système de la sonde. Elle porte son numéro de série et est accompagnée d'un **encart d'identification** qui indique l'édition acquise, le numéro de série du produit et le **code d'installation** (§ 6.4.3). Conservez cet encart en lieu sûr : le code d'installation donne accès à la mise en service. La référence du produit figure sur votre bon de commande ; la version de l'image est indiquée dans les notes de version et, après la mise en service, par `GET /api/v1/system` (champ `version`).

> ! **Attention** — Ne reformatez pas la microSD et n'y installez pas d'autre système. Si vous ouvrez la carte sur un ordinateur pour y déposer un fichier de provisionnement (§ 6.3), votre système d'exploitation peut proposer de « formater » une partition qu'il ne sait pas lire : **refusez systématiquement**. Seule la partition de dépôt, au format FAT32 et étiquetée `SVA-BOOT`, est destinée à être ouverte depuis un poste de travail. Avec le mécanisme de mise à jour A/B, d'autres partitions de la carte peuvent apparaître sur votre poste : ne les modifiez pas.

5.4 Assemblage avec la microSD préconfigurée

5.4.1 Matériel à réunir

Objectif : disposer de tous les éléments nécessaires avant de commencer l'assemblage.

Le tableau suivant liste le matériel que vous devez fournir. Les exigences détaillées figurent au chapitre 4.

Élément	Exigence	Remarque
Raspberry Pi 5	RAM selon l'édition (§ 5.3.2)	Modèle 5 exclusivement
Alimentation	Alimentation officielle Raspberry Pi 27 W USB-C (5,1 V, 5 A) recommandée	Une alimentation 5 V / 3 A fonctionne, mais limite le courant disponible sur les ports USB ; c'est un comportement documenté du Raspberry Pi 5
Refroidissement	Refroidisseur actif officiel (*Active Cooler*) ou boîtier équipé d'un ventilateur	**Obligatoire pour l'édition Enterprise**, recommandé pour les autres
Boîtier	Boîtier compatible Raspberry Pi 5, ventilé, laissant accès au lecteur microSD, au port Ethernet et au port USB-C	Le boîtier officiel Raspberry Pi 5 intègre un ventilateur
Câble Ethernet	Catégorie 5e minimum, longueur adaptée	Câble droit standard
microSD préconfigurée	Fournie	Ne pas la remplacer par une autre carte
Facultatif	Écran HDMI + câble micro-HDMI vers HDMI	Pour lire le code d'installation et l'empreinte du certificat à l'écran (§ 6.4.3)
Facultatif	Carte M.2 HAT+ officielle de Raspberry Pi et SSD NVMe M.2 2230 ou 2242 pris en charge	Volume de données complémentaire, surtout utile pour l'édition Enterprise (§ 4.3.3). Exige l'alimentation officielle de 27 W
Facultatif	Pile officielle de l'horloge temps réel (RTC) du Raspberry Pi 5	Recommandée si le port réseau est protégé par 802.1X : elle conserve l'heure hors tension, nécessaire à la vérification des certificats avant tout accès NTP (§ 4.5.1, § 7.2.6)

> ✓ **Conseil** — Le boîtier officiel Raspberry Pi 5 et le refroidisseur actif officiel ne s'utilisent généralement pas ensemble, le boîtier disposant de son propre ventilateur. Suivez la notice du fabricant de votre boîtier pour savoir quel refroidissement installer. Si vous installez la carte M.2 HAT+ et un SSD (§ 5.4.6), le boîtier officiel n'est utilisable que **sans son couvercle ni son ventilateur** : le refroidissement est alors assuré par le refroidisseur actif officiel, installé en premier.

5.4.2 Préparer la microSD (facultatif)

Si vous avez choisi la mise en service par **fichier de provisionnement** (méthode A du chapitre 6), préparez la microSD **maintenant**, avant de l'insérer dans la carte. La procédure est décrite au § 6.3. Si le port réseau de la sonde est protégé par 802.1X et que la sonde doit s'y authentifier dès son premier démarrage, déposez à cette étape, à côté du fichier de provisionnement, le fichier PKCS#12 de la sonde et le certificat de l'AC du serveur RADIUS (§ 6.3.3 et § 7.2.6). Si vous utilisez l'assistant web (méthode B), passez directement au § 5.4.3.

5.4.3 Installer le refroidissement

Objectif : assurer une dissipation thermique suffisante pour un fonctionnement permanent.

Prérequis : carte hors tension et hors de son sachet ; précautions ESD appliquées (§ 5.2.1).

Étapes (refroidisseur actif officiel) :

1. Posez la carte sur une surface plane, face supérieure visible.
2. Retirez les films de protection des pastilles thermiques situées sous le refroidisseur.
3. Positionnez le refroidisseur au-dessus de la carte en alignant ses deux tiges de fixation sur les deux trous de montage prévus à cet effet.
4. Enfoncez les tiges de fixation en appuyant **sur les tiges elles-mêmes**, et non sur le corps du refroidisseur, jusqu'au clic.
5. Branchez le câble du ventilateur sur le connecteur marqué « FAN », en respectant le détrompeur.
6. Vérifiez que le câble ne touche pas les pales du ventilateur.

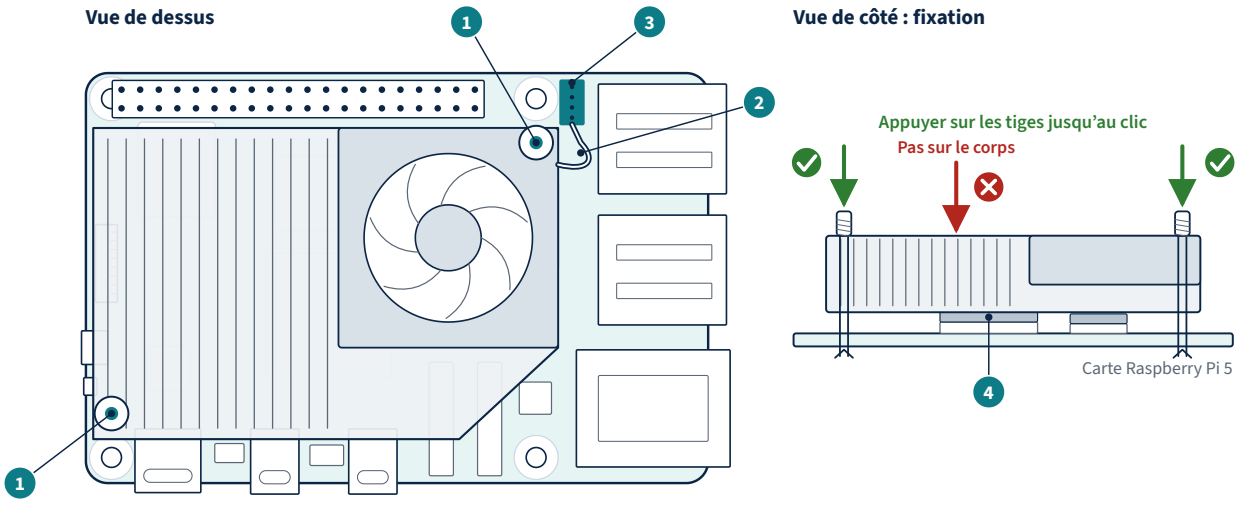


Figure 5-3 — Pose du refroidisseur actif officiel : alignement des tiges de fixation et branchement sur le connecteur FAN

Résultat attendu : le refroidisseur est fermement maintenu, sans jeu ; le câble est branché.

Vérification : la mise en rotation du ventilateur sera observée au premier démarrage. Le ventilateur peut ne pas tourner en permanence : sa vitesse est pilotée selon la température.

Avec un boîtier ventilé : suivez la notice du fabricant ; raccordez son ventilateur au connecteur « FAN » s'il est prévu pour cela.

> **Attention** — Un refroidisseur ne se repose pas : les pastilles thermiques perdent leur efficacité une fois décollées. Ne retirez le refroidisseur qu'en cas de nécessité.

5.4.4 Installer la carte dans le boîtier

Objectif : protéger la carte et faciliter son installation.

Prérequis : refroidissement installé (§ 5.4.3) ; si vous avez retenu l'option SSD, carte M.2 HAT+ et SSD déjà montés (§ 5.4.6).

Étapes :

1. Ouvrez le boîtier selon la notice du fabricant. Suivez aussi ses éventuelles instructions propres au refroidissement (dissipateur fourni avec le boîtier, par exemple), sans les cumuler avec le refroidisseur actif officiel si la notice l'exclut.
2. Placez la carte dans le fond du boîtier en alignant les ports avec les ouvertures correspondantes.
3. Fixez la carte si le boîtier le prévoit (vis ou ergots), sans serrage excessif.
4. Si le boîtier est équipé d'un ventilateur, raccordez-le au connecteur « FAN ».
5. **Ne refermez pas encore le boîtier** s'il masque le lecteur microSD : certains boîtiers exigent l'insertion de la carte avant fermeture. Dans ce cas, réalisez d'abord le § 5.4.5.
6. Refermez le boîtier.

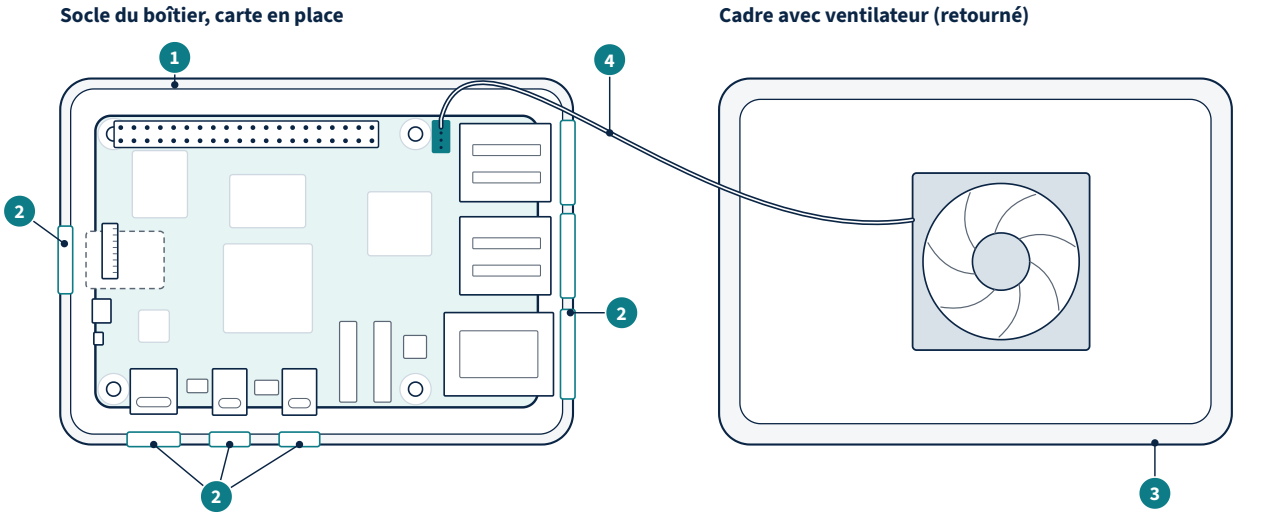


Figure 5-4 — Carte installée dans le boîtier officiel Raspberry Pi 5 : alignement des ports et raccordement du ventilateur

Résultat attendu : la carte est maintenue ; le port USB-C, le port Ethernet et le lecteur microSD restent accessibles.

5.4.5 Insérer la microSD

Objectif : installer le support contenant le système de la sonde.

Prérequis : carte **hors tension** ; microSD préparée si vous utilisez la méthode A (§ 6.3).

Étapes :

1. Repérez le lecteur microSD sur la **face inférieure** de la carte, sur le petit côté opposé aux ports Ethernet et USB (figure 5-2).
2. Tenez la microSD par ses bords, **contacts dorés tournés vers la carte électronique** : lorsque vous regardez la face inférieure de la carte retournée, c'est l'étiquette de la microSD qui est visible.
3. Faites glisser la microSD dans le lecteur, sans forcer, jusqu'à ce qu'elle soit entièrement insérée. Le lecteur du Raspberry Pi 5 n'a pas de mécanisme à ressort : la carte est maintenue par friction.
4. Vérifiez que la carte ne dépasse que légèrement du bord de la carte électronique ou de l'ouverture du boîtier.

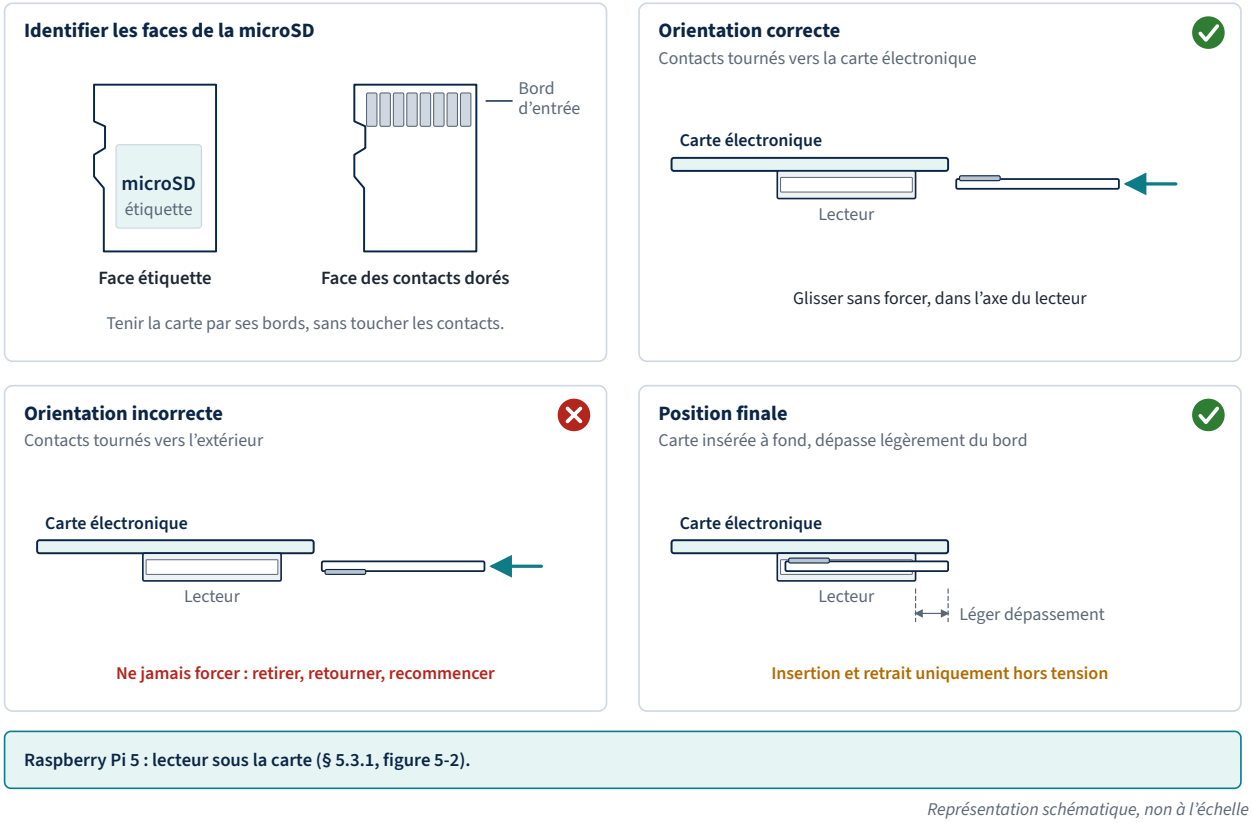


Figure 5-5 — Insertion de la microSD : orientation correcte (contacts vers la carte) et position finale

Résultat attendu : la microSD est insérée à fond et ne bouge pas lorsqu'on effleure son bord.

En cas de problème : si la carte n'entre pas, retirez-la, vérifiez son orientation et recommencez. Ne forcez jamais.

> ! **Attention** — N'insérez et ne retirez jamais la microSD lorsque la sonde est alimentée. Un retrait à chaud peut corrompre le système de fichiers et rendre la sonde inutilisable.

5.4.6 Monter la carte M.2 HAT+ et le SSD NVMe (facultatif)

Cette opération ne concerne que les sondes équipées du SSD NVMe facultatif (§ 4.3.3). Dans l'ordre d'assemblage, elle se place **après** la pose du refroidisseur actif (§ 5.4.3) et **avant** la mise en boîtier (§ 5.4.4). Elle se limite au montage physique : la sonde n'utilise le SSD qu'après son activation, décrite au chapitre 20 (§ 20.3.4).

Objectif : raccorder un SSD NVMe au Raspberry Pi 5 par la carte M.2 HAT+ officielle.

Prérequis :

- carte hors tension, câble d'alimentation USB-C débranché ; précautions ESD appliquées (§ 5.2.1) ;
- refroidisseur actif officiel déjà installé, s'il est utilisé (§ 5.4.3) ;
- carte M.2 HAT+ officielle de Raspberry Pi, avec les accessoires de montage et la notice fournis par le fabricant ;
- SSD NVMe au format M.2 2230 ou 2242, figurant dans la liste des modèles pris en charge (§ 4.3.3) ;
- alimentation officielle Raspberry Pi 27 W USB-C, qui sera utilisée au § 5.8.

Étapes :

1. Vérifiez que le refroidisseur actif, si vous l'utilisez, est déjà en place : il ne peut pas être posé après la carte M.2 HAT+.
2. Installez les entretoises fournies avec la carte M.2 HAT+, ainsi que l'embase GPIO empilable si elle est fournie, selon la notice du fabricant.
3. Raccordez la nappe PCIe au connecteur PCIe du Raspberry Pi 5 (connecteur plat, repère sur l'figure 5-10) : soulevez délicatement le verrou du connecteur, insérez la nappe à fond dans l'orientation indiquée par la notice, puis rabattez le verrou.
4. Placez la carte M.2 HAT+ sur les entretoises et fixez-la avec les vis fournies, sans serrage excessif.
5. Raccordez l'autre extrémité de la nappe au connecteur de la carte M.2 HAT+ : verrou soulevé, nappe insérée à fond, verrou rabattu. L'orientation des contacts de ce côté est celle qu'indique la notice du M.2 HAT+ ; elle n'est pas nécessairement la même qu'à l'étape précédente.
6. Insérez le SSD dans l'emplacement M.2 de la carte, en biais, contacts en premier, puis abaissez-le et fixez-le avec la vis prévue, à la position qui correspond à sa longueur (2230 ou 2242).
7. Vérifiez que la nappe n'est pas pliée à angle vif et qu'elle ne touche pas les pales du ventilateur.
8. Poursuivez avec la mise en boîtier (§ 5.4.4). Avec le boîtier officiel Raspberry Pi 5, n'installez ni son couvercle ni son ventilateur.

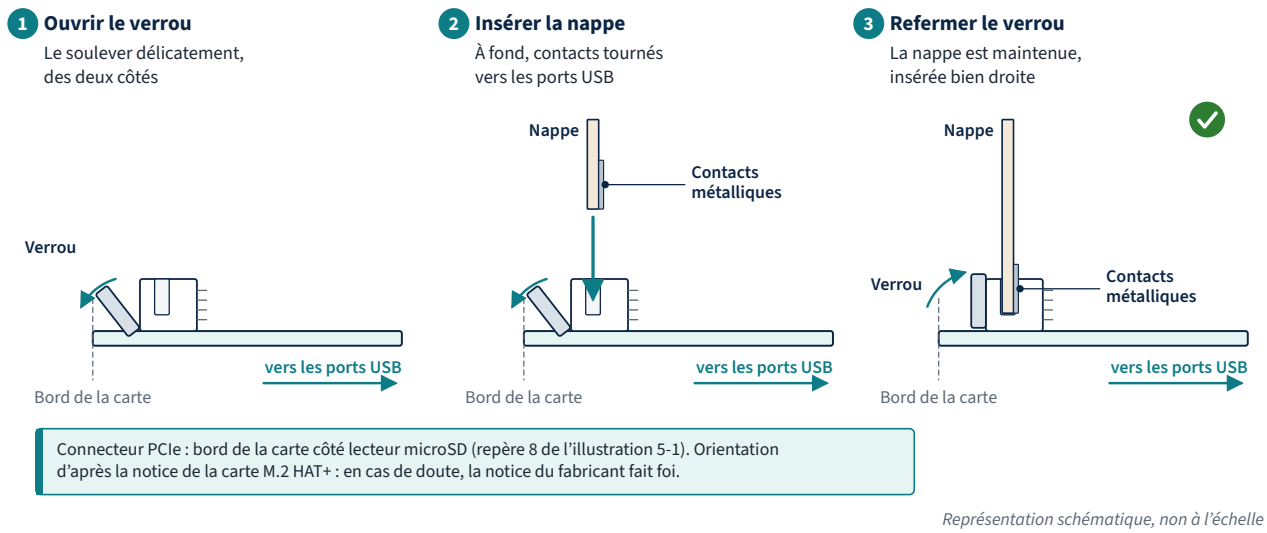


Figure 5-10 — Raccordement de la nappe PCIe au connecteur PCIe du Raspberry Pi 5 : verrou ouvert, orientation de la nappe, verrou refermé

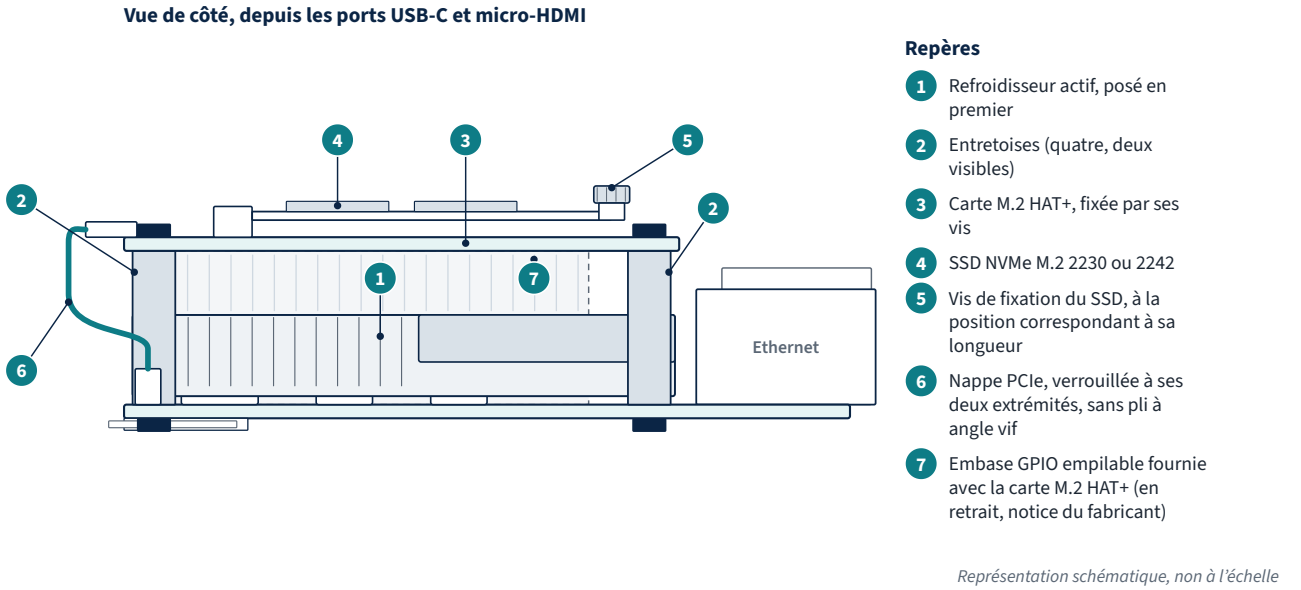


Figure 5-11 — Ensemble monté : refroidisseur actif, carte M.2 HAT+ sur ses entretoises, SSD fixé par sa vis

Résultat attendu : la carte M.2 HAT+ est fixée sans jeu ; la nappe est verrouillée à ses deux extrémités ; le SSD est inséré à fond et maintenu par sa vis.

Vérification : contrôle visuel à ce stade. La détection du SSD par la sonde se vérifie lors de son activation (chapitre 20, § 20.3.4).

Erreurs possibles : si le SSD n'est pas détecté lors de l'activation, les causes habituelles sont une nappe mal insérée ou inversée, un SSD absent de la liste des modèles pris en charge ou une alimentation non conforme. Arrêtez proprement la sonde, débranchez l'alimentation, puis vérifiez le montage.

Précautions :

- La nappe PCIe est fragile : ne la pliez pas à angle vif et ne tirez pas dessus pour la retirer ; ouvrez toujours le verrou du connecteur.
- Ne modifiez aucun fichier système de la microSD, en particulier la configuration de démarrage du Raspberry Pi (`config.txt`), ni pour activer le mode PCIe Gen 3, ni pour changer l'ordre de démarrage : seule la partition `SVA-BOOT` reçoit vos fichiers, et uniquement ceux que décrit cette documentation ; la sonde n'utilise pas le mode Gen 3 et démarre toujours sur la microSD.
- N'installez aucun système sur le SSD et n'y copiez pas l'image de la sonde : le système, la configuration et les secrets restent sur la microSD.
- L'activation du SSD le **formate entièrement** : n'utilisez pas un SSD contenant des données à conserver.

> **Information** — Dans l'offre boîtier complet, la présence d'un SSD dépend de votre commande : reportez-vous à la liste de colisage.

5.5 Mise en place du boîtier complet

5.5.1 Déballage

Objectif : sortir le matériel sans l'endommager et vérifier son état.

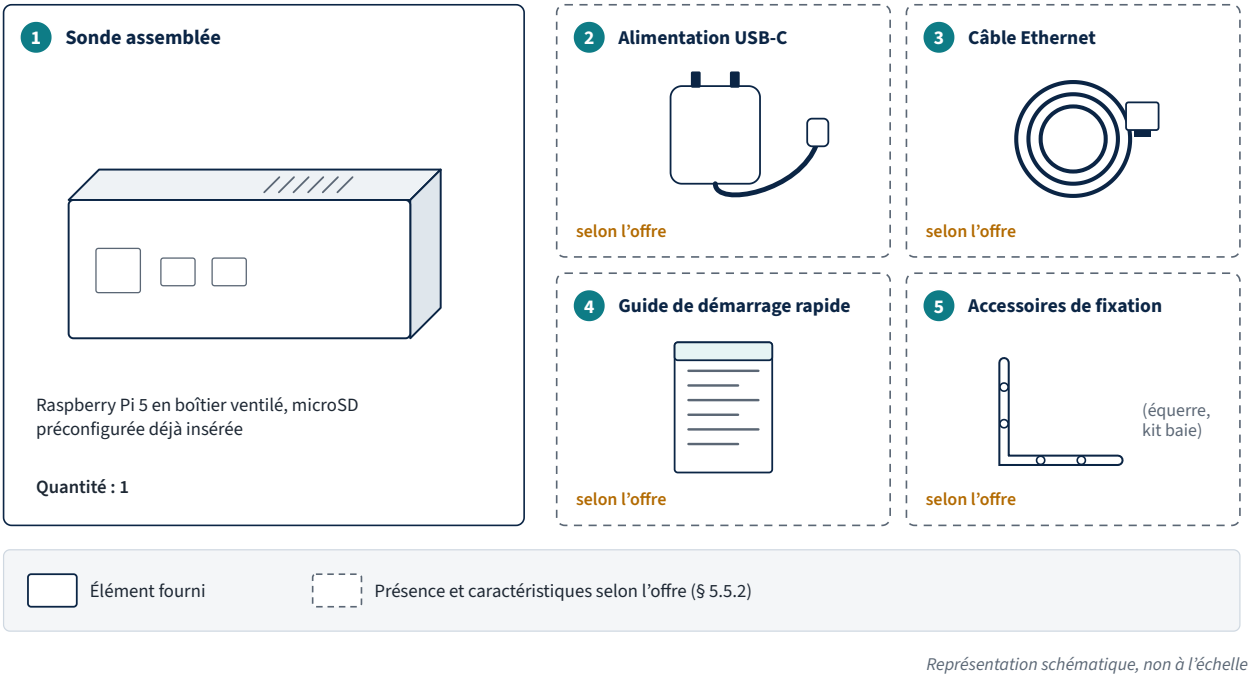
Étapes :

1. Inspectez l'emballage extérieur. En cas de choc visible ou d'emballage endommagé, photographiez-le avant ouverture et signalez-le au transporteur.
2. Ouvrez le colis et sortez les éléments un par un.
3. Conservez l'emballage d'origine : il est recommandé pour tout retour de matériel.
4. Contrôlez visuellement le boîtier : absence de fissure, de pièce mobile à l'intérieur (bruit en inclinant légèrement le boîtier), de connecteur endommagé.

5.5.2 Contenu du colis

Offre boîtier complet.

Élément	Quantité	Vérifié
Sonde assemblée (Raspberry Pi 5 en boîtier ventilé, microSD préconfigurée insérée)	1	[ ]
Alimentation USB-C	Selon la liste de colisage de votre commande ; à défaut, voir § 4.2	[ ]
Câble Ethernet	Selon la liste de colisage de votre commande ; à défaut, à fournir par vous (§ 4.5.1)	[ ]
Guide de démarrage rapide	Selon la liste de colisage de votre commande	[ ]
Accessoires de fixation (équerre, kit baie)	Selon la liste de colisage de votre commande	[ ]



Représentation schématique, non à l'échelle

Figure 5-6 — Contenu du colis de l'offre boîtier complet

Offre microSD préconfigurée (§ 5.4). Le colis contient les éléments suivants ; tout autre élément est fourni par vos soins (§ 5.4.1).

Élément	Quantité	Vérifié
Carte microSD contenant le système et le logiciel de la sonde, portant son numéro de série (§ 5.3.3)	1	[ ]
Encart d'identification : édition acquise, numéro de série du produit et code d'installation (§ 5.3.3)	1	[ ]

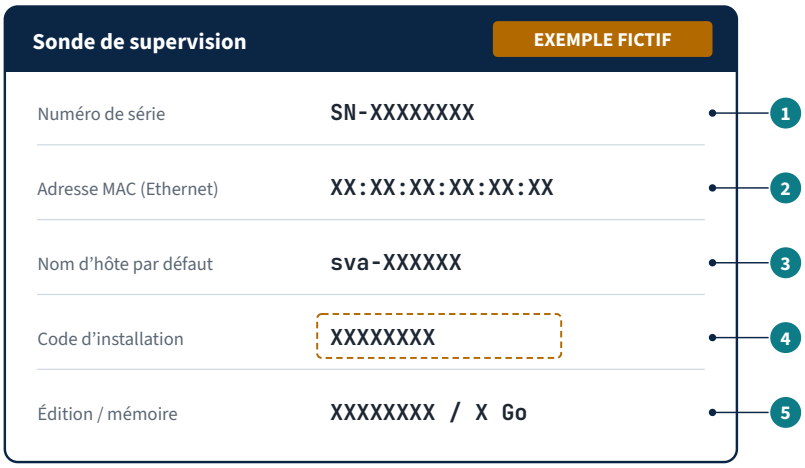
En cas d'élément manquant ou endommagé, contactez le support avant toute installation (coordonnées au chapitre 21).

> **Information** — Dans l'offre boîtier complet, la microSD est déjà insérée. Il n'est pas nécessaire d'ouvrir le boîtier. Si vous souhaitez utiliser un fichier de provisionnement (§ 6.3), retirez la microSD selon les indications jointes au boîtier livré ou, à défaut, communiquées par le support ; ne forcez jamais sur le lecteur.

5.5.3 Étiquette d'identification

Le boîtier porte une étiquette d'identification. Relevez ses informations **avant** l'installation, car l'étiquette peut devenir difficile d'accès une fois la sonde placée en baie.

Information	Usage
Numéro de série	Identification auprès du support
Adresse MAC de l'interface Ethernet	Réservation DHCP, recherche de la sonde sur le réseau (§ 6.4.2)
Nom d'hôte par défaut	De la forme `sva-XXXXXX`, où `XXXXXX` correspond aux six derniers caractères hexadécimaux de l'adresse MAC
Code d'installation	Accès à l'assistant de mise en service (§ 6.4.3)
Édition et capacité mémoire	Vérification de la conformité de la commande



Champs

1. Numéro de série : identification auprès du support
2. Adresse MAC : réservation DHCP, recherche sur le réseau
3. Nom d'hôte par défaut : sva- suivi des six derniers caractères de l'adresse MAC
4. Code d'installation : à protéger comme un mot de passe
5. Édition et capacité mémoire : conformité de la commande

Valeurs fictives : présentation indicative de l'étiquette.

Représentation schématique, non à l'échelle

Figure 5-7 — Exemple d'étiquette d'identification du boîtier complet, avec repérage de chaque champ

> **Avertissement** — Le code d'installation permet de prendre le contrôle d'une sonde non encore configurée. Ne le diffusez pas et ne le photographiez pas sur un support partagé. Il ne permet plus d'ouvrir l'assistant après la mise en service, mais il reste exigé pour récupérer le compte administrateur (§ 20.10.2) et redevient valide après une réinitialisation usine : conservez l'encart ou l'étiquette pendant toute la vie du produit.

> **Conseil** — Recopiez l'adresse MAC et le numéro de série dans la fiche de mise en service (chapitre 21) et transmettez l'adresse MAC à l'équipe réseau si une réservation DHCP est souhaitée (§ 7.2).

5.6 Choix de l'emplacement

Objectif : installer la sonde dans un lieu qui garantit sa fiabilité, sa sécurité et la pertinence de ses mesures.

5.6.1 Critères techniques

Critère	Recommandation
Position réseau	La sonde mesure ce qu'elle « voit » depuis son emplacement. Placez-la sur le segment réseau dont vous voulez évaluer l'expérience : réseau des utilisateurs pour mesurer l'accès à une application, réseau du centre de données pour superviser des services internes. Voir § 7.10
Sécurité physique	Local fermé, baie verrouillée ou local technique. La microSD contient la configuration et les secrets chiffrés de la sonde : un accès physique non contrôlé constitue un risque (§ 7.11)
Ventilation	Espace libre d'au moins quelques centimètres autour des ouïes d'aération ; pas d'empilement contre un équipement chaud
Alimentation	Prise secourue (onduleur) recommandée : une coupure électrique brutale est l'une des principales causes d'usure et de corruption des cartes microSD
Câblage	Prise réseau ou port de commutateur disponible à proximité ; longueur de câble Ethernet inférieure à 100 m
Accès	Accès possible pour la maintenance (retrait de la microSD, observation du voyant)

5.6.2 Installation en baie

La sonde n'est pas un équipement au format rack 19 pouces. Pour une installation en baie, posez-la sur une **tablette** fixe ou utilisez un **support** adapté, notamment l'accessoire de montage éventuellement prévu par votre commande. Ne la posez pas directement sur le capot d'un serveur, dont la chaleur dégraderait son refroidissement.

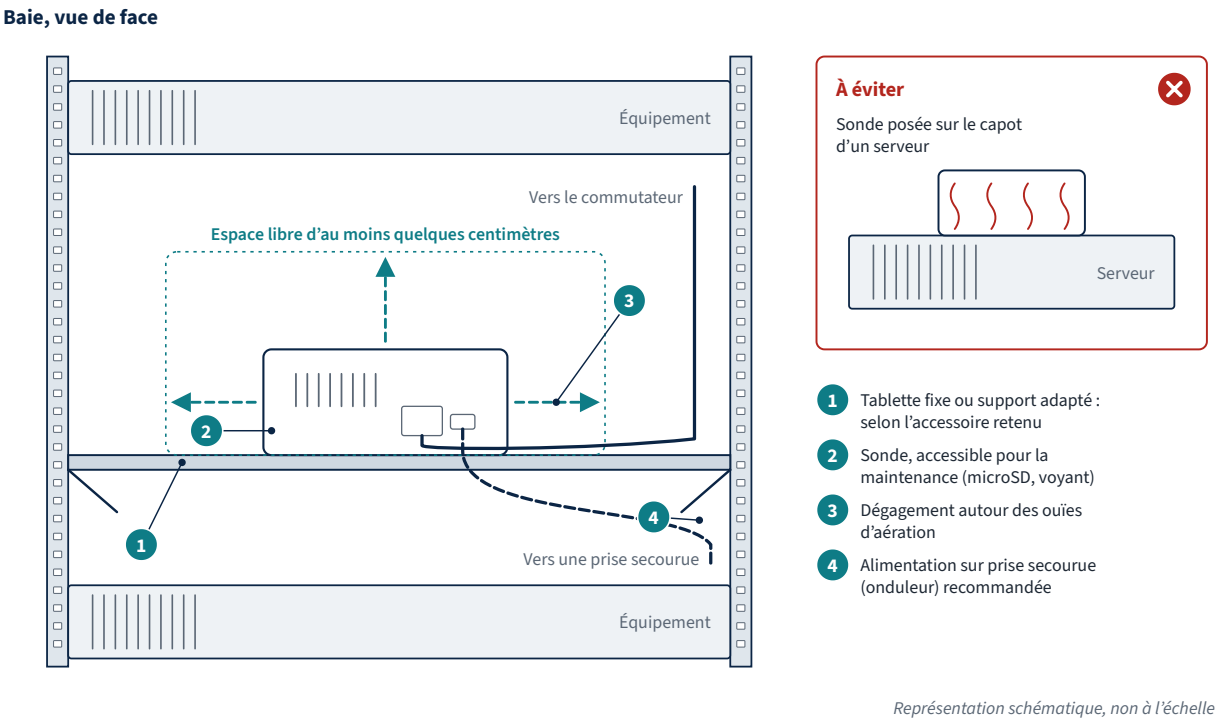


Figure 5-8 — Exemple d'installation sur tablette en baie, avec dégagement pour la ventilation

5.7 Connexion Ethernet

Objectif : raccorder la sonde au réseau choisi pour son implantation (§ 4.5.2).

Prérequis :

- port de commutateur ou prise murale active, sur le réseau prévu (VLAN de supervision, par exemple, voir § 7.10) ;
- si le port est protégé par le contrôle d'accès 802.1X, la sonde s'authentifie par certificat (EAP-TLS). Deux possibilités, à arrêter avec l'équipe réseau (§ 4.5.1 et § 7.2.6) : le fichier PKCS#12 de la sonde et le certificat de l'AC du serveur RADIUS ont été déposés sur la microSD avec le fichier de provisionnement (§ 5.4.2), ou le port est provisoirement non contrôlé ou autorisé par adresse MAC (MAB) pour la mise en service, 802.1X étant activé ensuite ;
- si la sonde doit obtenir une adresse par DHCP, un serveur DHCP doit être actif sur ce réseau ; sinon, une configuration statique doit être fournie par le fichier de provisionnement (§ 6.3).

Étapes :

1. Branchez une extrémité du câble Ethernet sur le port RJ45 de la sonde, jusqu'au clic du loquet.
2. Branchez l'autre extrémité sur le port du commutateur ou la prise murale.
3. Ne branchez pas encore l'alimentation.

Résultat attendu : le câble est verrouillé aux deux extrémités.

Vérification : les voyants du port Ethernet de la sonde et du commutateur s'allumeront après la mise sous tension (§ 5.9.2).

> **Information** — La sonde fonctionne exclusivement en Ethernet. Le Wi-Fi et le Bluetooth intégrés au Raspberry Pi 5 sont désactivés dans le système de la sonde, par une surcouche de l'arbre des périphériques (*device tree overlay*).

→ **Évolution prévue** — La prise en charge du Wi-Fi est envisagée dans une version ultérieure.

5.8 Branchement de l'alimentation et démarrage

Objectif : mettre la sonde sous tension dans de bonnes conditions.

Prérequis : microSD insérée ; boîtier fermé ; câble Ethernet branché ; méthode de mise en service choisie (chapitre 6).

Étapes :

1. Branchez l'alimentation sur la prise secteur (de préférence secourue).
2. Branchez le connecteur USB-C sur le port d'alimentation de la sonde. **L'alimentation doit être branchée en dernier.**
3. La sonde démarre automatiquement dès qu'elle est alimentée. Il n'est normalement pas nécessaire d'appuyer sur le bouton d'alimentation.
4. Ne débranchez pas la sonde pendant le premier démarrage, qui comporte des opérations d'initialisation (extension du système de fichiers, génération des clés de sécurité). Sa durée est indiquée au § 6.2.

Résultat attendu : le voyant d'état s'allume, puis traduit l'activité du système (§ 5.9.1).

! **Attention** — Ne coupez jamais l'alimentation d'une sonde en fonctionnement en débranchant le câble, sauf en dernier recours. Utilisez la procédure d'arrêt propre décrite au chapitre 20. Une coupure brutale peut provoquer une perte des dernières données et, de manière répétée, une corruption de la microSD.

i **Information** — Si l'alimentation utilisée ne fournit pas le courant attendu par le Raspberry Pi 5 (alimentation 5 V / 3 A, par exemple), la carte peut fonctionner avec un courant USB limité. La sonde surveille les situations de sous-tension et les signale dans ses indicateurs de santé (chapitres 17 et 19).

5.9 Vérifications initiales

5.9.1 Voyant d'état

Le Raspberry Pi 5 dispose d'un voyant d'état unique, bicolore. D'après la documentation du fabricant, sa couleur indique l'état d'alimentation de la carte :

Observation	Signification (comportement général du Raspberry Pi 5)	Action
Aucun voyant allumé	Cause probable : la carte n'est pas alimentée (à recouper avec l'écran HDMI éventuel et les voyants Ethernet, encadré ci-dessous)	Vérifier l'alimentation, le câble et la prise (§ 5.10)
Rouge fixe	La carte est alimentée mais le système est arrêté ou en veille	Appuyer brièvement sur le bouton d'alimentation pour démarrer ; si la situation persiste, voir § 5.10
Vert, clignotant de façon irrégulière	Le système fonctionne ; le clignotement traduit l'activité de lecture et d'écriture	Situation normale
Clignotements répétitifs selon un motif régulier (séquences de flashs longs et courts)	Code d'erreur du chargeur d'amorçage, documenté par Raspberry Pi (par exemple, microSD illisible ou système introuvable)	Noter le motif, consulter la documentation Raspberry Pi et le chapitre 19

La sonde ne définit aucune signalisation propre : le voyant ne présente que les comportements standard du Raspberry Pi 5 décrits ci-dessus. En particulier, il n'indique ni la fin de l'initialisation ni l'état de configuration de la sonde ; ces informations se vérifient sur l'écran HDMI éventuel ou par le réseau (chapitre 6).

> **Information** — Les couleurs et clignotements du voyant sont des **indices matériels** : leur comportement exact dépend du microprogramme (*firmware*) du Raspberry Pi 5, de sa configuration et de l'étape du démarrage. Un voyant vert actif signale une activité de lecture et d'écriture sur le support de stockage ; il ne confirme ni que le logiciel de la sonde fonctionne, ni qu'il est correctement configuré. Pour établir l'état de la sonde, recoupez toujours le voyant avec la console de l'écran HDMI éventuellement raccordé (adresse IP, empreinte du certificat, état de la mise en service) et, dès que la sonde est joignable, avec l'adresse `/health` (§ 6.6.2).

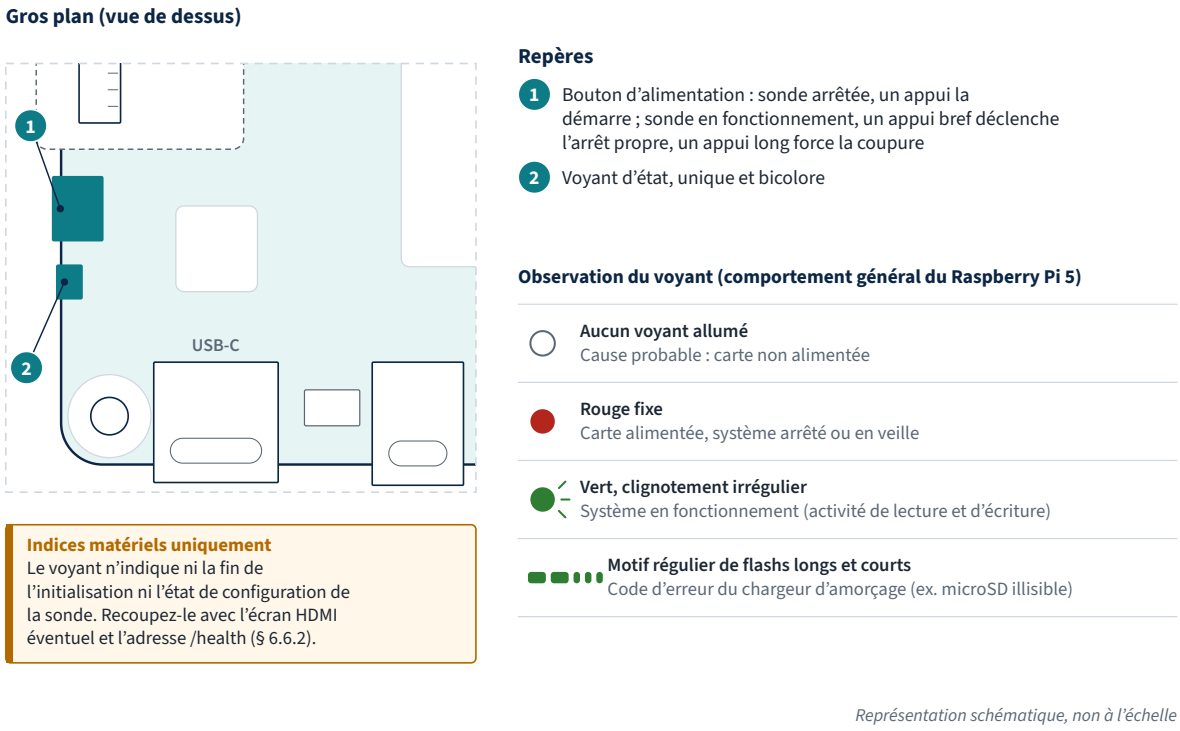


Figure 5-9 — Emplacement du voyant d'état et du bouton d'alimentation du Raspberry Pi 5

5.9.2 Voyants du port Ethernet

Le port Ethernet comporte deux voyants intégrés :

Observation	Signification
Voyant de liaison allumé	Une liaison physique est établie avec le commutateur
Voyant d'activité clignotant	Des trames circulent sur le lien
Aucun voyant	Pas de liaison : câble, port de commutateur ou prise murale à vérifier

Vérifiez également, côté commutateur, que le port est actif et affecté au bon VLAN. L'équipe réseau peut confirmer qu'une adresse MAC correspondant à la sonde est apprise sur le port.

5.9.3 Ventilateur

Le ventilateur peut tourner brièvement à pleine vitesse au démarrage, puis ralentir ou s'arrêter selon la température. Un ventilateur qui ne tourne jamais, alors que la sonde est très chaude au toucher, doit être vérifié (branchement sur le connecteur « FAN »).

5.9.4 Liste de vérification de fin d'installation physique

Point de contrôle	Vérifié
Matériel conforme à l'édition (RAM, refroidissement)	[ ]
Refroidissement installé et raccordé	[ ]
microSD insérée à fond, sonde hors tension lors de l'insertion	[ ]
Emplacement ventilé, sécurisé, alimentation secourue si possible	[ ]
Câble Ethernet verrouillé, voyants de liaison allumés	[ ]
Alimentation branchée en dernier	[ ]
Voyant d'état vert avec activité	[ ]
Adresse MAC et numéro de série relevés	[ ]
Si SSD prévu : carte M.2 HAT+ et SSD montés, nappe verrouillée, alimentation de 27 W utilisée (§ 5.4.6)	[ ]

L'installation physique est terminée. Poursuivez avec la mise en service au chapitre 6.

5.10 Erreurs fréquentes lors de l'installation

Symptôme	Causes possibles	Vérifications et actions
Aucun voyant	Alimentation non branchée ou défectueuse ; prise sans courant	Tester la prise avec un autre appareil ; essayer une autre alimentation conforme
Voyant rouge fixe, pas de démarrage	Système arrêté ; alimentation insuffisante	Appui bref sur le bouton d'alimentation ; utiliser l'alimentation recommandée (§ 5.4.1)
Motif de clignotement répétitif	microSD absente, mal insérée, endommagée ou ne contenant pas l'image de la sonde	Mettre hors tension, réinsérer la microSD à fond ; vérifier qu'il s'agit bien de la carte fournie ; voir chapitre 19
Voyant vert actif mais pas de voyant Ethernet	Câble non verrouillé ou défectueux ; port de commutateur inactif	Changer de câble ; tester un autre port ; vérifier la configuration du port avec l'équipe réseau
Ventilateur bruyant en permanence	Température élevée ; emplacement mal ventilé	Améliorer la ventilation ; vérifier l'absence de source de chaleur proche
La sonde redémarre de façon répétée	Alimentation insuffisante ; microSD défectueuse	Utiliser l'alimentation recommandée ; contacter le support si le problème persiste

Symptôme	Causes possibles	Vérifications et actions
Voyant de liaison Ethernet allumé, mais la sonde n'obtient pas d'adresse, sur un port protégé par 802.1X	Authentification 802.1X refusée, ou sonde sans identifiant client	Consulter l'état affiché sur l'écran HDMI éventuel et le journal du serveur RADIUS ; voir § 7.2.6 et chapitre 19
Aucun démarrage, avec ou sans message sur l'écran HDMI, sur une carte autre qu'un Raspberry Pi 5	Modèle de carte non pris en charge (Raspberry Pi 4, Raspberry Pi 400, Compute Module, Orange Pi…)	Le système vérifie le modèle au tout début du démarrage, avant tout montage en écriture et avant l'extension de la partition ; s'il a pu démarrer, il s'arrête sans monter de partition en écriture et affiche un message sur l'écran HDMI. Si le microprogramme de la carte ne charge pas l'image, aucun composant de la sonde ne s'exécute et aucun message ne peut s'afficher. Dans les deux cas, le contrôle vise à préserver la microSD, sans que sa conservation puisse être garantie lorsque le microprogramme d'une autre carte a manipulé le support (§ 4.1.1). Mettez hors tension et installez la carte dans un Raspberry Pi 5 (§ 4.1.1).
La microSD a été « formatée » par un ordinateur	Acceptation d'une proposition de formatage du système d'exploitation du poste	Le formatage a effacé le système et le certificat d'édition, qui ne peut pas être réinscrit par vos soins : la carte doit être remplacée. Contactez le support avec le numéro de série du produit (chapitre 20, § 20.6.4)

Le diagnostic détaillé de ces situations est traité au chapitre 19.

Chapitre 6 — Premier démarrage et mise en service

Ce chapitre décrit les opérations qui transforment une sonde neuve, tout juste raccordée, en sonde opérationnelle : initialisation du système, configuration initiale, vérification de l'édition, vérifications et exécution d'un premier contrôle.

À l'issue de ce chapitre, la sonde :

- dispose d'une adresse IP connue et d'un nom d'hôte ;
- est protégée par un compte administrateur qui vous est propre ;
- a produit un code de récupération de ce compte, que vous avez enregistré dans le coffre-fort de votre organisation ;
- présente un certificat dont vous avez vérifié l'empreinte ;
- a son édition vérifiée ;
- a exécuté un premier contrôle dont le résultat est consultable.

> **Information** — Les exemples de ce chapitre utilisent des adresses et des noms fictifs, réservés à la documentation : réseau `192.0.2.0/24`, domaine `example.internal`, sonde `sonde-paris-01`. Remplacez-les par les valeurs de votre environnement.

6.1 Les deux méthodes de mise en service

La sonde ne contient **aucun mot de passe par défaut**. Tant qu'elle n'est pas configurée, elle n'accepte qu'une seule forme d'accès protégée. Deux méthodes sont proposées.



Offre boîtier complet : déballer et contrôler le contenu (§ 5.5) remplace l'étape 1.
Méthode A : le fichier provision.yaml est déposé sur la microSD avant l'étape 1 et appliqué au premier démarrage (§ 6.3).
Étape 5 : comparer l'empreinte affichée par le navigateur à celle de l'écran HDMI ou du fichier setup-code.txt, avant toute saisie (§ 6.4.4).

Figure 6-10 — Les étapes de la mise en service, de l'insertion de la microSD au premier contrôle

	Méthode A — Fichier de provisionnement	Méthode B — Assistant web
Principe	Un fichier `provision.yaml`, préparé sur un poste de travail, est déposé sur la microSD avant le premier démarrage. La sonde l'applique automatiquement	Après le démarrage, un assistant web, protégé par un code d'installation, guide la configuration

	Méthode A — Fichier de provisionnement	Méthode B — Assistant web
Adaptée à	Déploiements en série ; réseaux sans DHCP (adresse statique dès le premier démarrage) ; ports soumis à l'authentification 802.1X ; automatisation	Installation unitaire ; utilisateurs préférant une interface guidée
Nécessite	Un poste capable de lire une microSD ; l'empreinte d'un mot de passe (§ 6.3.4)	Un serveur DHCP sur le réseau de la sonde ; l'accès au code d'installation ; un écran micro-HDMI ou un lecteur de microSD pour vérifier l'empreinte du certificat (§ 6.4.4)
Intervention après démarrage	Vérification, puis relevé du code de récupération à la première connexion (§ 6.4.6)	Parcours de l'assistant (durée indicative : une dizaine de minutes, selon les options choisies)

✓ **Conseil** — Sur un réseau sans serveur DHCP, utilisez la méthode A : sans adresse IP, la sonde ne pourrait pas être jointe par l'assistant web. Il en va de même sur un port de commutateur soumis à l'authentification 802.1X : seule la méthode A permet de fournir à la sonde son identifiant 802.1X avant tout accès au réseau (§ 6.3.3). À défaut, effectuez la mise en service sur un port non contrôlé ou autorisé par adresse MAC, puis activez 802.1X (chapitre 7).

Les deux méthodes aboutissent au même résultat. Un fichier de provisionnement valide suffit à configurer la sonde : les sections qu'il ne contient pas prennent leurs valeurs par défaut (§ 6.3.3). Si le fichier est refusé, la sonde reste non initialisée et la mise en service peut se poursuivre avec l'assistant.

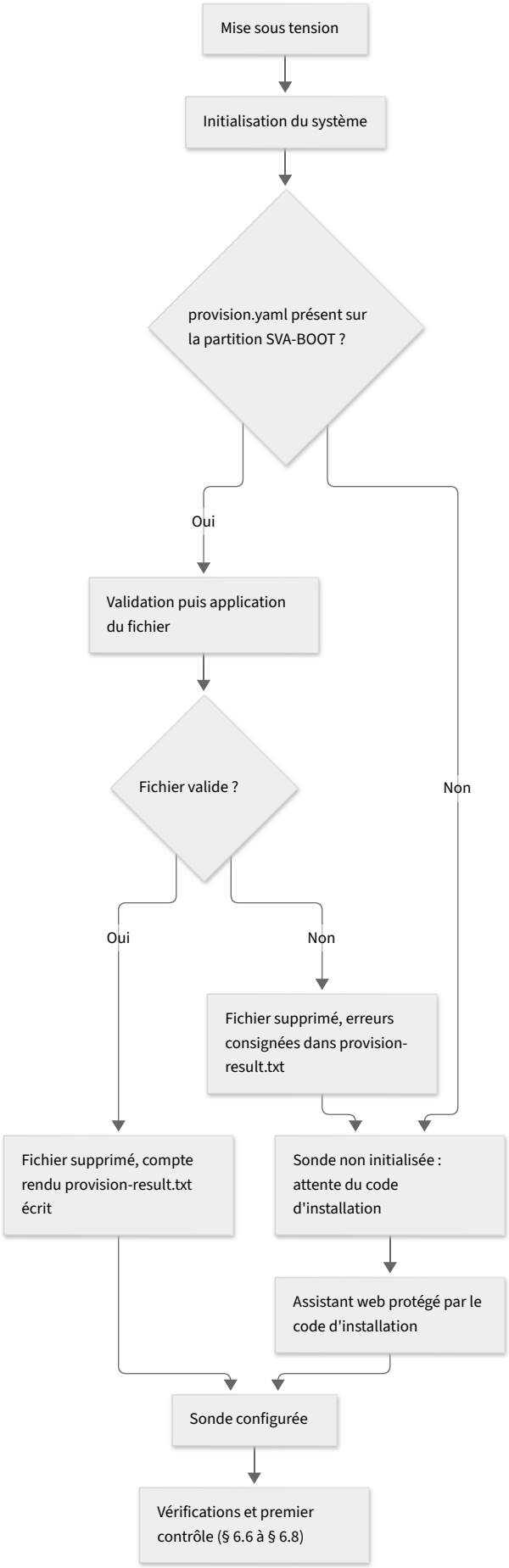


Schéma 6-1 — Premier démarrage et choix de la méthode de mise en service

> **Attention** — La validation du fichier est **atomique** : soit le fichier est entièrement valide et tous ses paramètres sont appliqués, soit aucun ne l'est. Dans les deux cas, le fichier est supprimé, car il contient des données sensibles, et le motif d'un refus est consigné, sans aucun secret, dans `provision-result.txt`. Consultez ce compte rendu après chaque provisionnement.

6.2 Déroulement de l'initialisation

Au premier démarrage, la sonde exécute des opérations qui ne se reproduiront pas lors des démarrages suivants :

Ordre	Opération	Effet visible
1	Démarrage du système Raspberry Pi OS	Voyant vert, activité intense
2	Extension de la partition de données à la capacité restante de la microSD (les emplacements système A/B, en lecture seule, ne sont pas modifiés)	Activité disque soutenue
3	Génération des identifiants propres à l'appareil : identifiant unique de sonde, clés SSH (même si SSH reste désactivé), clé privée et certificat TLS autosigné, clé de chiffrement des secrets	Aucun
4	Lecture du fichier de provisionnement s'il est présent	Écriture de `provision-result.txt`
5	Si le fichier de provisionnement contient une section `dot1x` : authentification 802.1X (EAP-TLS) sur la prise Ethernet, **avant** la configuration réseau. Puis configuration réseau : DHCP par défaut, ou paramètres du provisionnement ; IPv6 seulement s'il est activé	Voyants Ethernet actifs ; requête DHCP visible côté serveur. En cas d'échec de l'authentification 802.1X : nouvelles tentatives, état affiché sur la console HDMI
6	Démarrage du moteur principal et du moteur de scénarios	Interface d'administration accessible
7	Si la sonde n'est pas configurée : attente de la mise en service	Code d'installation (inscrit sur la carte lors de sa préparation) et empreinte du certificat affichés sur l'écran HDMI éventuel, avec l'adresse IP et le nom mDNS ; code et empreinte écrits dans `setup-code.txt`. Contrairement aux autres opérations du tableau, l'écriture de ce fichier se répète à chaque démarrage ; le code en disparaît une fois la sonde initialisée (encadré ci-dessous)

> **Information** — Le fichier `setup-code.txt` de la partition `SVA-BOOT` n'est pas propre à l'attente de mise en service : la sonde le **réécrit à chaque démarrage**, qu'elle soit configurée ou non, avec l'empreinte SHA-256 du certificat qu'elle présente à ce moment. Le code d'installation n'y figure que tant que la sonde n'est pas initialisée ; une fois la mise en service terminée, le fichier ne contient plus que l'empreinte. De même, la console de l'écran HDMI affiche l'empreinte courante à chaque démarrage. Ces deux canaux, indépendants du réseau, sont donc disponibles quel que soit le parcours de mise en service, y compris le provisionnement sans écran (§ 6.3).

Le premier démarrage est plus long que les suivants, en raison de l'extension du système de fichiers et de la génération des clés. À titre indicatif, comptez quelques minutes. Les démarrages suivants sont plus rapides.

> **Attention** — Ne coupez pas l'alimentation pendant le premier démarrage. Une interruption pendant l'extension du système de fichiers ou la génération des clés peut rendre la carte inutilisable ; elle se remplace alors selon le § 20.6.4.

> **Information** — Les clés et le certificat sont générés **sur la sonde**, au premier démarrage. Deux sondes issues de la même image ne partagent donc aucun secret. C'est aussi la raison pour laquelle il ne faut pas dupliquer une microSD déjà mise en service pour en équiper une autre sonde (voir chapitre 20).

6.3 Mise en service par fichier de provisionnement (méthode A)

6.3.1 Principe

La microSD comporte une petite **partition de dépôt** au format FAT32, étiquetée `SVA-BOOT`, lisible depuis Windows, macOS et Linux ; dans cette documentation, l'expression « partition de démarrage » désigne cette partition. Avec le mécanisme de mise à jour A/B, d'autres partitions peuvent apparaître sur votre poste : ne les modifiez pas. Si un fichier nommé `provision.yaml` est présent à la racine de cette partition au démarrage d'une sonde non encore initialisée, la sonde :

1. le lit et valide son contenu ;
2. applique l'ensemble de ses paramètres si le fichier est entièrement valide, et aucun dans le cas contraire (validation atomique) ;
3. **supprime** le fichier dans tous les cas, afin qu'aucune information sensible ne reste lisible sur la carte ;
4. écrit, au même emplacement, un compte rendu `provision-result.txt` qui ne contient aucun secret.

Le format YAML (*YAML Ain't Markup Language*) est un format texte structuré par l'indentation. Il est lisible et modifiable avec tout éditeur de texte.

6.3.2 Prérequis

- Un poste de travail équipé d'un lecteur de cartes microSD (ou d'un adaptateur USB).
- Un éditeur de texte brut (Bloc-notes, TextEdit en mode texte brut, Visual Studio Code, vi…). **N'utilisez pas** un traitement de texte, qui ajouterait une mise en forme.
- Les paramètres réseau de la sonde si l'adresse doit être statique : adresse et masque, passerelle, serveurs DNS, serveurs NTP.
- L'empreinte du mot de passe administrateur (§ 6.3.4).
- Si l'IPv6 doit être activé en adressage statique : adresse IPv6 et longueur de préfixe, passerelle IPv6.
- Si le port de commutateur est soumis à l'authentification 802.1X : l'identité à présenter, l'identifiant client de la sonde sous forme de fichier PKCS#12 et son mot de passe, ainsi que le certificat, au format PEM, de l'autorité de certification (AC) du serveur RADIUS. Ces éléments sont fournis par l'équipe en charge du contrôle d'accès réseau.

6.3.3 Référence des champs

Le tableau suivant décrit les champs reconnus par la version 1 du format.

Champ	Obligatoire	Description	Exemple
`version`	Oui	Version du format du fichier ; vaut `1`	`1`

Champ	Obligatoire	Description	Exemple
`hostname`	Non	Nom d'hôte de la sonde : lettres minuscules, chiffres et tirets, 63 caractères maximum. Par défaut : `sva-XXXXXX`	`sonde-paris-01`
`site`	Non, recommandé	Emplacement logique de la sonde ; repris comme attribut de toutes les données exportées (chapitre 18)	`paris-dc1`
`admin.username`	Oui	Identifiant du premier compte administrateur	`admin`
`admin.passwordHash`	Oui	**Empreinte** du mot de passe, jamais le mot de passe en clair (§ 6.3.4)	—
`network.mode`	Non	`dhcp` (par défaut) ou `static`	`static`
`network.address`	Si `static`	Adresse IPv4 et longueur de préfixe	`192.0.2.50/24`
`network.gateway`	Si `static`	Passerelle par défaut	`192.0.2.1`
`network.dns`	Si `static`	Liste de serveurs DNS (deux recommandés), en IPv4 ou en IPv6	`[192.0.2.10, 192.0.2.11]`
`network.ntp`	Non, recommandé	Liste de serveurs NTP. Par défaut : serveurs fournis par DHCP (option 42), sinon une zone fournisseur du *NTP Pool*, indiquée dans les notes de version	`[ntp1.example.internal]`
`network.ipv6.enabled`	Non	Active l'IPv6, en complément de l'IPv4 qui reste toujours actif. Par défaut : `false`	`true`
`network.ipv6.mode`	Si l'IPv6 est activé	`slaac` (configuration automatique par les annonces du routeur) ou `static`. Valeur par défaut : `slaac`	`static`
`network.ipv6.address`	Si `ipv6.mode` vaut `static`	Adresse IPv6 et longueur de préfixe (notation CIDR)	`2001:db8:10::50/64`
`network.ipv6.gateway`	Si `ipv6.mode` vaut `static`	Passerelle IPv6 par défaut	`2001:db8:10::1`
`proxy.url`	Non	Proxy HTTP(S) sortant	`http://proxy.example.internal:3128`

Champ	Obligatoire	Description	Exemple
`proxy.noProxy`	Non	Destinations joignables sans proxy : suffixes de domaine, adresses, plages	`[".example.internal", "192.0.2.0/24"]`
`dot1x.identity`	Si la section `dot1x` est présente	Identité présentée au serveur RADIUS lors de l'authentification 802.1X (EAP-TLS) sur la prise Ethernet. La présence de la section `dot1x` déclenche l'authentification avant tout accès au réseau	`sonde-paris-01.example.internal`
`dot1x.pkcs12File`	Si la section `dot1x` est présente	Nom du fichier PKCS#12 (clé privée et certificat client de la sonde), déposé à la racine de la partition `SVA-BOOT`. Supprimé après traitement	`sonde-paris-01.p12`
`dot1x.caFile`	Si la section `dot1x` est présente	Nom du fichier contenant le certificat de l'AC du serveur RADIUS, au format PEM, déposé à la racine de la partition `SVA-BOOT`. Supprimé après traitement	`radius-ca.pem`
`dot1x.pkcs12Password`	Si la section `dot1x` est présente	Mot de passe du fichier PKCS#12. **Seul secret admis dans ce fichier**, supprimé avec lui après traitement	—
`ssh.locked`	Non	`true` verrouille **définitivement** l'accès SSH : il ne pourra plus être activé, jusqu'à une réinitialisation usine (§ 7.9.4). Par défaut : `false`	`true`
`updates.security.mode`	Non, recommandé	Mode du canal des correctifs de sécurité : `automatic` (par défaut) ou `manual`. En mode `automatic`, les correctifs sont téléchargés et installés sur l'emplacement inactif dès leur disponibilité, puis activés dans une fenêtre de maintenance	`automatic`

Champ	Obligatoire	Description	Exemple
`updates.security.criticalPolicy`	Non	Activation d'un correctif de sécurité critique (vulnérabilité activement exploitée) : `next-window` (par défaut : au début de la première fenêtre qui commence au moins 15 minutes après l'annonce) ou `asap` (choix explicite : 15 minutes après l'annonce, augmentées de `criticalStagger`, même hors fenêtre)	`next-window`
`updates.security.criticalStagger`	Non	Décalage fixe, propre à la sonde, ajouté au préavis de 15 minutes d'un correctif critique en mode `asap`, de `0` à `4h`, pour échelonner les sondes d'un même site (chapitre 20, § 20.4.11)	`0`
`updates.feature.mode`	Non	Mode du canal des versions fonctionnelles : `manual` (par défaut) ou `automatic`	`manual`
`updates.windows`	Non, recommandé	Liste de 0 à 7 fenêtres de maintenance, chacune avec `days` (jours parmi `mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun`), `start` (heure de début au format `HH:MM`, **entre guillemets**) et `duration`. **Aucune fenêtre par défaut**	`[{days: [tue], start: "22:00", duration: 2h}]`
`updates.notice`	Non	Préavis minimal annoncé avant toute activation automatique, de `0` à `7d` (activation manuelle et correctif critique : chapitre 20, § 20.4.2). Par défaut : `24h`	`24h`
`updates.drainTimeout`	Non	Délai accordé aux exécutions en cours pour se terminer au début d'une activation, `30m` au maximum. Par défaut : `5m`	`5m`

Les champs ci-dessus constituent le format version 1, y compris les sections `dot1x`, `network.ipv6`, `ssh.locked` et `updates`.

Section `updates`. Elle fixe la politique de mises à jour dès le premier démarrage (chapitre 20). Une mise à jour est téléchargée et installée sur l'emplacement système inactif sans interrompre les contrôles ; son **activation**, c'est-à-dire le redémarrage sur la nouvelle version, n'a lieu que dans l'une des fenêtres que vous définissez. Si le fichier ne contient pas de section `updates`, la sonde applique les valeurs par défaut du tableau, **sans aucune fenêtre** : aucune mise à jour n'est alors activée automatiquement, et l'état de santé `updates` reste en `WARNING` tant qu'aucune fenêtre n'est définie. Un administrateur active alors chaque mise à jour par l'interface ou par l'API (`POST /api/v1/updates/apply`, chapitre 20). Aucune plage horaire n'est imposée : choisissez des créneaux compatibles avec vos plans d'exécution (chapitre 12).

Exemple de section `updates` (valeurs fictives, à ajouter au fichier de provisionnement) :

```yaml
# Politique de mises à jour — exemple fictif
updates:
  security:
    mode: automatic             # automatic (par défaut) | manual
    criticalPolicy: next-window # next-window (par défaut) | asap
    criticalStagger: "0"      # décalage des activations asap, de 0 à 4h
  feature:
    mode: manual                # manual (par défaut) | automatic
  windows:                      # aucune fenêtre par défaut ; 7 fenêtres au maximum
    - days: [tue, thu]
      start: "22:00"            # format HH:MM, toujours entre guillemets
      duration: 2h
  notice: 24h                   # préavis avant activation, de 0 à 7d
  drainTimeout: 5m              # 30m au maximum
```

> **Information** — Le choix des fenêtres et du préavis relève de votre organisation : le règlement européen sur la cyberrésilience (CRA) n'impose ni horaire ni calendrier particulier. Une sonde unique ne supervise pas pendant son propre redémarrage ; pour les environnements critiques, le chapitre 20 décrit le déploiement de plusieurs sondes par site avec des fenêtres qui ne se chevauchent pas. Les autres réglages (destinations d'export, authentification du proxy, certificat fourni par votre organisation, domaine de recherche DNS, activation temporaire de SSH) se configurent après la mise en service, dans l'interface ou par l'API : plusieurs d'entre eux font appel à des secrets, qui n'ont pas leur place dans ce fichier. Seule exception, le mot de passe du fichier PKCS#12 (`dot1x.pkcs12Password`) : il est indispensable pour que la sonde s'authentifie auprès du commutateur avant de disposer d'un accès au réseau.

> **Avertissement** — Ne placez **aucun secret applicatif** (mot de passe d'un compte de test, jeton d'une plateforme d'observabilité) dans le fichier de provisionnement. Même si le fichier est supprimé après lecture, il a pu être copié ou rester dans la corbeille du poste qui l'a préparé. Les secrets se saisissent dans l'interface d'administration ou par l'API (chapitre 9).

Avertissement — La section `dot1x` place sur la microSD le mot de passe du fichier PKCS#12, le fichier PKCS#12 lui-même, qui contient la clé privée de la sonde, et le certificat de l'AC. La sonde supprime ces fichiers et ce mot de passe après traitement, comme `provision.yaml`. Préparez-les sur un poste de confiance, supprimez-en toute copie (y compris dans la corbeille) une fois la carte préparée, et ne laissez pas la carte sans surveillance avant son insertion dans la sonde. Lorsque c'est possible, préférez une première mise en service sur un port non contrôlé ou autorisé par adresse MAC, suivie d'une génération de la clé et de la demande de certificat sur la sonde elle-même (chapitre 7) : la clé privée ne quitte alors jamais la sonde. La suppression des fichiers par la sonde ne vaut pas effacement physique de la mémoire flash : les précautions qui en découlent (certificat dédié et de courte durée, carte traitée comme un support sensible, révocation en cas de perte) sont détaillées au § 6.3.5.

Information — L'authentification 802.1X dès le premier démarrage repose sur la section `dot1x` du fichier de provisionnement : elle a lieu avant la configuration réseau et, en cas d'échec, la sonde renouvelle ses tentatives et affiche l'état de l'authentification sur la console HDMI (§ 6.2).

Attention — Le verrouillage de SSH (`ssh.locked: true`) est **irréversible** : seule une réinitialisation usine permet de le lever. Le support de l'éditeur ne pourra plus intervenir par SSH sur la sonde. La procédure de récupération du compte administrateur (chapitre 20, § 20.10.2) reste possible.

6.3.4 Empreinte du mot de passe administrateur

Une **empreinte** (ou *hash*) est le résultat d'une fonction cryptographique à sens unique appliquée au mot de passe. La sonde peut vérifier qu'un mot de passe saisi correspond à l'empreinte, mais l'empreinte ne permet pas de retrouver le mot de passe. Le fichier de provisionnement ne contient ainsi jamais le mot de passe lui-même.

La sonde accepte une empreinte calculée avec l'algorithme **Argon2id**, au format PHC (*Password Hashing Competition*), de la forme `$argon2id$v=19$m=…,t=…,p=…$<sel>$<empreinte>`. Toute bibliothèque standard d'Argon2 peut la produire.

L'exemple suivant utilise la bibliothèque publique Python `argon2-cffi`, installée dans un environnement virtuel sur le poste de préparation (Linux ou macOS ; sous Windows, remplacez `argon2-env/bin/` par `argon2-env\Scripts\`) :

```bash
# Créer un environnement virtuel et y installer la bibliothèque argon2-cffi
python3 -m venv argon2-env
argon2-env/bin/pip install argon2-cffi

# Saisir le mot de passe sans affichage, puis afficher son empreinte Argon2id au format PHC
argon2-env/bin/python -c "import getpass; from argon2 import PasswordHasher;
    print(PasswordHasher().hash(getpass.getpass('Mot de passe : ')))"
```

La commande affiche une ligne commençant par `$argon2id$v=19$` : c'est la valeur à reporter dans le champ `admin.passwordHash`, **entre guillemets** (§ 6.3.6). Le sel étant tiré au hasard, deux exécutions pour le même mot de passe produisent des empreintes différentes, toutes deux valides.

Règles à respecter :

1. Choisissez un mot de passe de 12 à 128 caractères, unique et non utilisé ailleurs. Aucune règle de composition n'est imposée : privilégiez une phrase de passe longue.
2. Générez l'empreinte sur un poste de confiance, avec l'outil indiqué ci-dessus.
3. Conservez le mot de passe dans le coffre-fort de mots de passe de votre organisation.

4. Ne transmettez jamais le mot de passe par messagerie ; seule l'empreinte figure dans le fichier.

6.3.5 Procédure

Objectif : configurer la sonde automatiquement dès son premier démarrage.

Prérequis : § 6.3.2 ; sonde **non encore initialisée** avec cette microSD (le fichier est lu à chaque démarrage tant que la sonde n'est pas initialisée, puis ignoré : une fois la sonde en service, sa configuration ne peut plus être écrasée par un fichier déposé sur la carte).

Étapes :

1. Insérez la microSD dans le lecteur de votre poste de travail.
2. Ouvrez la partition FAT32 étiquetée `SVA-BOOT` : c'est la seule partition de dépôt, qui reçoit tous les fichiers décrits dans cette documentation et ceux que produit la sonde. Avec le mécanisme de mise à jour A/B, d'autres partitions de la carte peuvent apparaître sur votre poste : **ne les modifiez pas**. Si la partition `SVA-BOOT` contient d'autres fichiers que ceux décrits dans cette documentation, **ne les modifiez pas et ne les supprimez pas**.
3. Si le système de votre poste propose de formater une autre partition illisible, **refusez** (§ 5.3.3).
4. Créez à la racine de cette partition un fichier texte nommé exactement `provision.yaml`. Sous Windows, vérifiez que l'extension `.txt` n'a pas été ajoutée automatiquement (affichez les extensions de fichiers dans l'explorateur).
5. Saisissez le contenu en vous appuyant sur l'exemple ci-dessous et sur le tableau du § 6.3.3. Si vous utilisez la section `dot1x`, copiez aussi à la racine de la partition les deux fichiers désignés par `pkcs12File` et `caFile`.
6. Enregistrez le fichier en encodage **UTF-8**.
7. Vérifiez la syntaxe YAML (§ 6.3.6).
8. Éjectez proprement la microSD avec la fonction d'éjection de votre système, puis retirez-la.
9. Insérez la microSD dans la sonde (§ 5.4.5) et mettez la sonde sous tension (§ 5.8).

Exemple de fichier (valeurs fictives) pour une adresse statique :

```yaml
# Fichier de provisionnement — exemple fictif
version: 1
hostname: sonde-paris-01
site: paris-dc1
admin:
  username: admin
  # Empreinte du mot de passe (jamais le mot de passe en clair)
  passwordHash: "<empreinte Argon2id générée selon le § 6.3.4>"
network:
  mode: static            # dhcp | static
  address: 192.0.2.50/24
  gateway: 192.0.2.1
  dns: [192.0.2.10, 192.0.2.11]
  ntp: [ntp1.example.internal]
proxy:
  url: http://proxy.example.internal:3128
  noProxy: [".example.internal", "192.0.2.0/24"]
```

Exemple minimal, en adressage DHCP :

```yaml
# Fichier de provisionnement minimal — exemple fictif
version: 1
hostname: sonde-lyon-01
site: lyon-siege
admin:
  username: admin
  passwordHash: "<empreinte Argon2id générée selon le § 6.3.4>"
```

Exemple pour un port soumis à l'authentification 802.1X, avec IPv6 activé et SSH verrouillé (valeurs fictives ; les fichiers `sonde-marseille-01.p12` et `radius-ca.pem` sont déposés à côté de `provision.yaml`) :

```yaml
# Fichier de provisionnement avec 802.1X et IPv6 — exemple fictif
version: 1
hostname: sonde-marseille-01
site: marseille-agence
admin:
  username: admin
  passwordHash: "<empreinte Argon2id générée selon le § 6.3.4>"
network:
  mode: dhcp
  ipv6:
    enabled: true          # IPv6 en complément de l'IPv4 (désactivé par défaut)
    mode: static           # slaac | static
    address: 2001:db8:10::50/64
    gateway: 2001:db8:10::1
dot1x:
  identity: sonde-marseille-01.example.internal
  pkcs12File: sonde-marseille-01.p12   # clé privée et certificat client de la sonde
  caFile: radius-ca.pem                # AC du serveur RADIUS, au format PEM
  # Seul secret admis dans ce fichier ; supprimé après traitement
  pkcs12Password: "<mot de passe du fichier PKCS#12>"
ssh:
  locked: true             # Irréversible jusqu'à une réinitialisation usine
```

Résultat attendu : après le premier démarrage, la sonde est joignable à l'adresse prévue et vous pouvez vous connecter à l'interface d'administration avec le compte défini. Lors de cette première connexion, l'interface affiche une seule fois le code de récupération du compte administrateur : enregistrez-le immédiatement (§ 6.4.6). Si une section `dot1x` a été fournie, `provision.yaml`, le fichier PKCS#12 et le certificat de l'AC ont disparu de la partition `SVA-BOOT`.

Vérification : § 6.3.7, puis § 6.6.

> **Attention — Vérifiez l'empreinte du certificat avant la première connexion**, selon la procédure du § 6.4.4 : le provisionnement s'achève sans passer par l'assistant, mais cette vérification reste obligatoire avant toute saisie d'identifiants et avant toute transmission d'un jeton d'API. Sans écran HDMI, relevez l'empreinte courante dans `setup-code.txt` lorsque vous retirez la microSD pour lire `provision-result.txt` (§ 6.3.7). Si les empreintes diffèrent, n'allez pas plus loin.

Avertissement — Effacement du fichier PKCS#12. La suppression du fichier PKCS#12, du certificat de l'AC et du mot de passe par la sonde est une suppression **logique** : sur une mémoire flash, elle ne garantit pas l'effacement physique des données, dont des fragments peuvent subsister sur la carte. Pour cette raison :

- préférez la voie qui génère la clé sur la sonde (voie CSR du § 7.2.6) chaque fois qu'un port non contrôlé ou autorisé par MAB peut être utilisé, même temporairement : la clé privée ne quitte alors jamais la sonde et ne transite jamais par la partition `SVA-BOOT` ;
- si la voie du provisionnement est indispensable, faites émettre pour la sonde un **certificat client dédié** à ce seul usage, de **courte durée de validité**, puis remplacez-le dès que possible par un certificat obtenu à partir d'une clé générée sur la sonde (renouvellement décrit au § 7.2.6) ;
- traitez la microSD comme un **support sensible** pendant toute sa vie, y compris lors de son remplacement ou de sa mise au rebut (effacement sécurisé, chapitre 20, § 20.13) ;
- en cas de perte ou de vol de la carte, faites **révoquer** sans délai le certificat client par votre IGC et retirez son autorisation du serveur RADIUS.

Attention — Heure de la sonde et EAP-TLS au premier démarrage. Sur un port 802.1X, la sonde n'accède au serveur NTP qu'après l'authentification, alors que la vérification des certificats exige une heure correcte. Les chemins de pré-réglage de l'horloge, sous votre contrôle, et leurs limites sont décrits au § 7.2.6. Si aucun d'eux n'est possible, la réussite de l'authentification EAP-TLS au premier démarrage ne peut pas être garantie.

6.3.6 Erreurs de syntaxe à éviter

Le YAML est sensible à l'indentation. Les erreurs les plus fréquentes sont les suivantes.

Erreur	Exemple incorrect	Correction
Tabulation au lieu d'espaces	Indentation par la touche Tab	Utiliser deux espaces par niveau
Indentation incohérente	`address` aligné différemment de `gateway`	Aligner les clés d'un même bloc
Espace manquant après les deux-points	`mode:static`	`mode: static`
Caractère spécial non protégé	Valeur contenant `#` ou `:` sans guillemets	Entourer la valeur de guillemets
Extension erronée	`provision.yaml.txt`	Renommer en `provision.yaml`
Heure non protégée	`start: 22:00` (peut être lu comme un nombre)	`start: "22:00"`

Conseil — Faites valider le fichier par un outil de vérification YAML (extension de votre éditeur, ou validateur en ligne de commande) avant de le copier sur la carte. N'utilisez pas de validateur en ligne public : ce fichier contient l'empreinte du mot de passe administrateur, la configuration de votre réseau et, le cas échéant, le mot de passe du fichier PKCS#12.

6.3.7 Lire le compte rendu `provision-result.txt`

Objectif : vérifier que le provisionnement a été appliqué intégralement.

Étapes :

1. Si la sonde est joignable, consultez de préférence le résultat dans le journal d'audit (interface d'administration, ou `GET /api/v1/audit-events`) : le provisionnement y est enregistré comme un événement, avec son résultat.

2. Sinon, arrêtez proprement la sonde (chapitre 20), retirez la microSD et ouvrez la partition `SVA-BOOT` sur votre poste.
3. Ouvrez `provision-result.txt`.

Le compte rendu est un fichier texte UTF-8 qui contient : la date et l'heure de traitement, le résultat global (appliqué ou refusé), la liste des sections appliquées et, en cas de refus, chaque erreur avec son numéro de ligne et son motif. Il ne contient aucune valeur sensible.

En cas de problème : corrigez les erreurs signalées. Si la sonde n'a pas été initialisée, elle attend une mise en service par l'assistant (méthode B) : vous pouvez soit poursuivre avec l'assistant, soit déposer un fichier corrigé sur la partition `SVA-BOOT` et redémarrer la sonde : le fichier est lu à chaque démarrage tant que la sonde n'est pas initialisée.

Si la sonde reste injoignable alors que le provisionnement comporte une section `dot1x`, l'authentification 802.1X a probablement échoué : la sonde réessaie et affiche l'état de l'authentification sur la console HDMI. Vérifiez avec l'équipe réseau l'identité, la validité du certificat client, l'AC du serveur RADIUS et la configuration du port (chapitre 19, § 19.5).

6.4 Mise en service par l'assistant web (méthode B)

La méthode B, décrite dans cette section, consiste à configurer la sonde après son démarrage, dans un assistant web protégé par le code d'installation. La méthode A désigne l'autre parcours : le provisionnement par fichier, décrit au § 6.3, où un fichier `provision.yaml` déposé sur la microSD configure la sonde dès son premier démarrage.

6.4.1 Prérequis

- Sonde installée et démarrée (chapitre 5), sans fichier de provisionnement.
- Serveur DHCP actif sur le réseau de la sonde.
- Port de commutateur sans authentification 802.1X, ou autorisé par adresse MAC (MAB, *MAC Authentication Bypass*) : l'assistant ne permet pas de fournir un identifiant 802.1X. Sur un port contrôlé, utilisez la méthode A, ou activez 802.1X après la mise en service (chapitre 7).
- Poste d'administration sur un réseau autorisé à joindre la sonde en HTTPS (port TCP 443), équipé d'un navigateur récent.
- Accès au code d'installation (§ 6.4.3).
- Paramètres à saisir : nom d'hôte et site souhaités, serveurs DNS et NTP, proxy éventuel ; l'adresse IP définitive est obtenue par une réservation DHCP, et non saisie dans l'assistant.

6.4.2 Trouver l'adresse IP de la sonde

La sonde obtient son adresse du serveur DHCP. Plusieurs moyens permettent de la retrouver.

> **Avertissement** — N'utilisez pas d'outil de balayage réseau (*scan*) sans l'accord de votre équipe sécurité : ce type d'outil peut déclencher des alertes de détection d'intrusion.

Moyen	Procédure	Remarque
Table des baux DHCP	Rechercher, dans la console du serveur DHCP, le bail associé à l'**adresse MAC** de la sonde (étiquette du boîtier complet) ou au nom d'hôte `sva-XXXXXX`	Méthode la plus fiable

Moyen	Procédure	Remarque
Réservation DHCP préalable	Créer avant le démarrage une réservation associant l'adresse MAC à une adresse choisie	Recommandé : l'adresse ne changera pas
Nom mDNS	Saisir `https://sva-XXXXXX.local/` dans le navigateur, `XXXXXX` correspondant aux six derniers caractères hexadécimaux de l'adresse MAC	Annonce mDNS active par défaut, désactivable dans la section `network` des paramètres. Fonctionne uniquement si le poste est sur le même segment réseau et si mDNS n'est pas filtré
Écran HDMI	Brancher un écran sur un port micro-HDMI : la console locale affiche l'adresse IP, le nom mDNS, l'empreinte du certificat et, tant que la sonde n'est pas initialisée, le code d'installation	Utile en l'absence d'accès au serveur DHCP
Table des adresses MAC du commutateur, table ARP du routeur	Rechercher l'adresse MAC sur le port de raccordement, puis l'adresse IP associée	Nécessite l'équipe réseau

Information — mDNS (*multicast DNS*) est un mécanisme de résolution de noms sans serveur, limité au réseau local. Les noms se terminent par `.local`. Il est souvent filtré entre VLAN.

6.4.3 Obtenir le code d'installation

Le code d'installation est un code propre à la carte, qui protège l'assistant : il est généré lors de la préparation de la carte et imprimé sur l'encart ou l'étiquette qui accompagne le produit. Il empêche qu'une autre personne présente sur le réseau ne s'approprie la sonde avant vous. Il devient invalide dès la fin de l'assistant.

Offre	Où trouver le code
Boîtier complet	Étiquette d'identification du boîtier (§ 5.5.3)
microSD préconfigurée	Encart d'identification qui accompagne la carte (§ 5.3.3)
Secours (les deux offres)	Écran branché en micro-HDMI au démarrage (console locale, avec l'empreinte du certificat), ou fichier `setup-code.txt` de la partition `SVA-BOOT`. Ce fichier contient aussi l'empreinte SHA-256 du certificat de la sonde, utile au § 6.4.4 ; il est réécrit à chaque démarrage avec l'empreinte courante (§ 6.2)

Si vous ne disposez plus de l'encart ou de l'étiquette, lisez `setup-code.txt` sans écran :

1. Laissez la sonde terminer son premier démarrage (§ 6.2).
2. Arrêtez-la proprement. Si l'interface n'est pas encore accessible, un appui bref sur le bouton d'alimentation déclenche l'arrêt propre (comportement standard de Raspberry Pi OS sur Raspberry Pi 5 ; un appui long force la coupure et doit être évité) ; attendez que le voyant passe au rouge fixe.
3. Débranchez l'alimentation, retirez la microSD et lisez le fichier `setup-code.txt` sur votre poste.
4. Réinsérez la microSD et redémarrez la sonde.

Le code d'installation reste valide d'un démarrage à l'autre tant que la sonde n'est pas initialisée : le code relevé reste donc valide après ce redémarrage. Il est invalidé à la fin de l'assistant, puis redevient valide après une réinitialisation usine (§ 20.10). Il est aussi exigé, avec le code de récupération, par la procédure de récupération du compte administrateur (§ 6.4.6) : conservez l'encart ou l'étiquette pendant toute la vie du produit.

Avertissement — Traitez le code d'installation comme un mot de passe. Toute personne qui le détient, et qui peut joindre la sonde sur le réseau, peut en prendre le contrôle tant que la mise en service n'est pas terminée. Effectuez la mise en service sans délai après le premier démarrage.

6.4.4 Se connecter et vérifier le certificat

La sonde présente un **certificat TLS autosigné**, généré à son premier démarrage. Un certificat autosigné n'est pas émis par une autorité de certification reconnue par votre navigateur : celui-ci affiche donc un avertissement de sécurité. Cet avertissement est attendu lors de la mise en service. Il ne doit cependant être accepté qu'après vérification de l'**empreinte** du certificat, qui garantit que vous êtes bien connecté à votre sonde et non à un équipement intercalé.

Cette section est la procédure de référence de la vérification d'empreinte, quel que soit le parcours de mise en service (méthode A ou B) et quel que soit l'outil utilisé ensuite (navigateur, `curl`, Prometheus, Nagios). Les autres sections de la documentation y renvoient.

Objectif : établir une connexion chiffrée et authentifiée avec la sonde.

Prérequis : l'empreinte SHA-256 courante du certificat, relevée par un canal indépendant du réseau (tableau ci-dessous).

Étapes :

1. Saisissez dans le navigateur `https://<adresse-ip-de-la-sonde>/` (exemple : `https://192.0.2.50/`).
2. Le navigateur signale que le certificat n'est pas reconnu. N'acceptez pas encore.
3. Affichez le détail du certificat (icône à gauche de la barre d'adresse, puis détail du certificat ; le libellé varie selon le navigateur). Relevez son **empreinte SHA-256**.
4. Comparez cette empreinte avec celle communiquée par la sonde par un canal indépendant du réseau : écran HDMI ou fichier `setup-code.txt` de la partition `SVA-BOOT` (§ 6.4.3). Seule cette comparaison hors réseau protège contre une interception : l'empreinte affichée ensuite par l'assistant ou par l'interface transite par la même connexion que celle que vous cherchez à vérifier.
5. Si les empreintes concordent, acceptez l'exception de sécurité pour cette adresse.
6. Si elles diffèrent, **interrompez la procédure** et contactez votre équipe sécurité : une interception de la connexion est possible.

> ! **Attention** — La vérification de l'empreinte est **obligatoire avant toute saisie d'identifiants** (code d'installation, compte administrateur) **et avant toute transmission d'un jeton d'API**. Elle se refait chaque fois que le certificat change (changement de nom ou d'adresse, renouvellement, réinitialisation usine), avec l'empreinte **courante** : une empreinte relevée avant un redémarrage ou un changement de certificat n'est plus une référence sûre.

Canal indépendant du réseau selon le parcours. L'empreinte courante est réécrite dans `setup-code.txt` à chaque démarrage et affichée sur la console HDMI (§ 6.2).

Parcours	Canal indépendant du réseau disponible
Provisionnement sans écran (méthode A)	Fichier `setup-code.txt`, lu en retirant la microSD après le premier démarrage (§ 6.4.3)
Assistant web (méthode B)	Console de l'écran HDMI, ou fichier `setup-code.txt` (§ 6.4.3)
Certificat émis par l'AC de votre entreprise (§ 7.7.2)	La chaîne de certification de votre entreprise, vérifiée par le navigateur ou par `curl` : aucune exception de sécurité n'est alors nécessaire
Après régénération ou renouvellement du certificat autosigné	Empreinte reçue par une connexion déjà vérifiée (§ 7.2.3) ; à défaut, console HDMI ou fichier `setup-code.txt`

Vérification en ligne de commande. Pour interroger la sonde avec `curl` ou un outil de supervision **sans désactiver la vérification TLS**, ces outils ont besoin du certificat de la sonde au format PEM. Récupérez-le, soit en l'exportant depuis votre navigateur après la vérification ci-dessus, soit par l'adresse `GET /api/v1/system/certificate`, puis vérifiez son empreinte avant de l'enregistrer, par exemple sous le nom `sonde-paris-01.pem` :

```bash
# Récupérer le certificat puis afficher son empreinte SHA-256 pour la comparer à celle de setup-
    code.txt
curl --insecure --silent https://192.0.2.50/api/v1/system/certificate -o sonde-paris-01.pem
openssl x509 -in sonde-paris-01.pem -noout -fingerprint -sha256
```

L'option `--insecure` n'est admise ici que pour ce téléchargement unique, effectué sans jeton et suivi obligatoirement de la comparaison d'empreinte. Supprimez le fichier si les empreintes diffèrent. Une alternative consiste à afficher l'empreinte du certificat présenté par le serveur avec OpenSSL :

```bash
# Afficher l'empreinte SHA-256 du certificat présenté par la sonde
openssl s_client -connect 192.0.2.50:443 </dev/null 2>/dev/null \
  | openssl x509 -noout -fingerprint -sha256
```

L'empreinte affichée doit être identique à celle relevée par le canal indépendant du réseau.

> **Information** — Le certificat autosigné couvre, dans son extension *Subject Alternative Name*, le nom d'hôte, le nom mDNS et l'adresse IPv4 de la sonde, ainsi que son adresse IPv6 statique lorsqu'elle est configurée. Si l'un de ces éléments change et qu'aucun certificat importé n'est en place, la sonde régénère son certificat : le fichier PEM enregistré n'est plus valable. Récupérez-le et vérifiez-le de nouveau comme ci-dessus.

> **Conseil** — À l'issue de la mise en service, il est recommandé de remplacer le certificat autosigné par un certificat émis par l'autorité de certification interne de votre organisation. Les navigateurs et outils de votre parc le reconnaîtront alors sans exception de sécurité (§ 7.7).

6.4.5 Parcourir l'assistant

L'assistant comporte les étapes suivantes, dans cet ordre. Les intitulés du tableau décrivent le contenu de chaque étape ; ils ne reproduisent pas nécessairement les libellés des écrans.

Étape	Contenu	Remarques
1. Code d'installation	Saisie du code (§ 6.4.3)	Après 5 saisies erronées, la saisie est bloquée pendant 15 minutes ; cette durée double à chaque nouvelle série d'échecs. Chaque échec est journalisé
2. Édition	Affichage de l'édition inscrite sur la carte (Essential, Professional, Enterprise) ; aucune saisie	Voir § 6.5
3. Compte administrateur	Identifiant et mot de passe du premier administrateur	Mot de passe de 12 à 128 caractères, sans règle de composition ni expiration périodique
4. Réseau	Nom d'hôte, site, mode DHCP ou statique, adresse, passerelle, DNS ; activation facultative de l'IPv6 (désactivé par défaut), en mode SLAAC ou statique (adresse IPv6, passerelle IPv6)	Ne changez pas l'adresse IP dans l'assistant ; utilisez une réservation DHCP (§ 7.2.1). Un changement d'adresse ou de nom d'hôte impose une réauthentification à la nouvelle adresse dans les 120 secondes (procédure après le tableau). L'IPv4 reste toujours actif ; les serveurs DNS peuvent être IPv4 ou IPv6

Étape	Contenu	Remarques
5. Heure	Serveurs NTP, fuseau horaire d'affichage	La sonde vérifie la synchronisation (§ 7.4)
6. Proxy	Proxy sortant éventuel et exclusions	Facultatif
7. Certificat	Affichage de l'empreinte SHA-256 ; option d'import d'un certificat de votre AC	L'import peut être réalisé plus tard (§ 7.7)
8. Destinations d'export	Configuration facultative d'une première destination (Prometheus, OTLP…)	Peut être reportée au chapitre 18
9. Mises à jour	Choix du mode de chaque canal (correctifs de sécurité : `automatic` ou `manual` ; versions fonctionnelles : `manual` ou `automatic`), politique des correctifs critiques, puis définition d'**au moins une fenêtre de maintenance** (jours, heure de début, durée), ou choix explicite du **mode manuel** (chaque activation est alors décidée par un administrateur) ; préavis et délai de drainage	**Aucune fenêtre n'est proposée par défaut** : l'étape ne peut être validée sans un choix explicite. Sans fenêtre, aucune mise à jour n'est activée automatiquement et l'état de santé `updates` reste en `WARNING` (chapitre 20)
10. Récapitulatif	Relecture et validation	Le code d'installation est invalidé à la validation
11. Code de récupération	Affichage **unique** du code de récupération du compte administrateur	À enregistrer immédiatement dans le coffre-fort de l'organisation (§ 6.4.6) ; il ne pourra plus être affiché

Résultat attendu : la sonde affiche la page d'accueil de l'interface d'administration ; vous êtes connecté avec le compte administrateur créé.

> ! **Attention** — Ne changez pas l'adresse IP dans l'assistant ; utilisez une réservation DHCP (§ 7.2.1). Si un changement d'adresse ou de nom d'hôte est indispensable à l'étape 4, vous devez vous réauthentifier à la nouvelle adresse dans les **120 secondes**, faute de quoi la sonde revient automatiquement à la configuration précédente : suivez la procédure ci-dessous.

Changement d'adresse ou de nom d'hôte à l'étape 4 (déconseillé). Après validation de l'étape 4, la sonde devient joignable à sa nouvelle adresse. La session d'administration est liée à l'adresse d'origine (cookie de session et protection contre la falsification de requêtes, CSRF) : à la nouvelle adresse, le navigateur ouvre une **nouvelle session**, et la confirmation n'est émise qu'après votre réauthentification. L'interface émet alors pour vous la confirmation `POST /api/v1/settings/network/confirm` décrite au § 7.2.3. Procédez ainsi :

1. Dans les 120 secondes, ouvrez la nouvelle adresse. Pendant ce délai, la sonde présente encore l'**ancien certificat**, celui dont vous avez vérifié l'empreinte au § 6.4.4 : le navigateur affiche la **même empreinte**, que vous reconnaissez avant d'accepter l'exception pour cette adresse.
2. Authentifiez-vous de nouveau : la confirmation est alors émise.
3. Lorsque le nom d'hôte ou l'adresse change, le certificat autosigné est régénéré pour les inclure, et son empreinte change. Après la confirmation, l'assistant affiche l'empreinte SHA-256 du nouveau certificat, reçue par la connexion déjà vérifiée, qui en fait une source de confiance ; le nouveau certificat est mis en service 30 secondes plus tard. Relevez cette empreinte.
4. Après la bascule, comparez l'empreinte présentée par le navigateur à celle relevée à l'étape 3, avant d'accepter l'exception. Aucune vérification hors réseau n'est nécessaire dans ce délai.

Sans confirmation, la sonde revient d'elle-même à la configuration précédente au bout de 120 secondes. La nouvelle empreinte s'affiche aussi sur la console HDMI, figure dans le journal d'audit et est écrite dans le fichier `setup-code.txt` de la partition `SVA-BOOT` au plus tard au démarrage suivant ; jusqu'à ce démarrage, ce fichier peut donc encore contenir l'empreinte précédente. Si la politique de votre navigateur interdit l'exception de sécurité, si la

réponse de confirmation est perdue ou si vous ne savez pas si la confirmation a été enregistrée, appliquez les recours du § 7.2.3. Un certificat émis par l'AC de votre entreprise ne dispense de l'exception que s'il couvre le nom ou l'adresse que vous utilisez pour joindre la sonde après le changement (§ 7.7.2).

6.4.6 Conserver le code de récupération du compte administrateur

Le **code de récupération** permet de redéfinir le mot de passe d'un compte administrateur si plus aucun administrateur ne peut se connecter, sans perdre la configuration de la sonde. Il comporte 24 caractères et est généré à la fin de la mise en service, quelle que soit la méthode :

- méthode B : il est affiché à la dernière étape de l'assistant (§ 6.4.5) ;
- méthode A : il est affiché à la première connexion à l'interface d'administration d'une sonde provisionnée.

La sonde ne conserve qu'une empreinte Argon2id de ce code (§ 6.3.4) : elle ne peut donc plus jamais l'afficher. La procédure de récupération exige à la fois un accès physique à la carte microSD, le code d'installation imprimé (§ 6.4.3) et ce code de récupération. Elle est décrite au chapitre 20, § 20.10.

Objectif : conserver le code de récupération de façon sûre et durable.

Prérequis : accès en écriture au coffre-fort de mots de passe de votre organisation, ouvert sur le poste d'administration avant la fin de la mise en service.

Étapes :

1. Lorsque le code de récupération s'affiche, ne quittez pas l'écran.
2. Enregistrez le code dans le coffre-fort de votre organisation, dans une entrée qui identifie la sonde (nom d'hôte, numéro de série).
3. Relisez l'entrée enregistrée et comparez-la au code affiché, caractère par caractère.
4. Quittez l'écran seulement ensuite.

Résultat attendu : le code de récupération est enregistré dans le coffre-fort et nulle part ailleurs.

Vérification : l'entrée du coffre-fort contient un code de 24 caractères, rattaché à la bonne sonde.

Erreurs possibles :

Situation	Conséquence	Action
Écran fermé avant l'enregistrement du code	Le code ne peut plus être affiché	Un administrateur génère un nouveau code depuis l'interface ou par `POST /api/v1/system/recovery-code` (rôle `admin`, saisie de son mot de passe exigée) : le nouveau code s'affiche une seule fois et l'ancien est invalidé. Tant qu'aucun code n'est disponible, la perte du dernier compte administrateur n'est réparable que par une réinitialisation usine (§ 20.10)
Code consommé par une récupération	Un nouveau code est affiché à l'issue de la récupération	Remplacer l'ancien code par le nouveau dans le coffre-fort

> **Avertissement** — Le code de récupération donne, avec le code d'installation et un accès physique à la carte, le contrôle d'un compte administrateur. Ne le conservez ni sur l'encart du produit, ni sur la sonde, ni dans un courriel. Ne le transmettez jamais au support de l'éditeur.

6.5 Édition de la sonde

L'édition détermine les types de contrôles disponibles.

Édition	Types de contrôles disponibles
Essential	ICMP, TCP, DNS, HTTP/HTTPS (y compris vérification du contenu), plugins Nagios fournis
Professional	Essential + suites de requêtes API, contrôles de bases de données et indicateurs métier, plugins Nagios personnalisés
Enterprise	Professional + scénarios navigateur (déclaratifs et scriptés) avec Puppeteer et Chromium

L'édition acquise est inscrite sur la carte microSD lors de sa préparation, sous la forme d'un **certificat d'édition** : un fichier signé par l'éditeur et lié à l'identifiant matériel de cette carte. La sonde le vérifie **localement** à chaque démarrage, sans connexion à un serveur de l'éditeur. Il n'y a ni clé à saisir, ni activation en ligne, ni date d'expiration : le droit d'utilisation du logiciel intégré est perpétuel. Seule une carte de démonstration, prêtée par l'éditeur pour une évaluation, est limitée à 30 jours (annexe A). Le certificat d'édition ne change pas lors des mises à jour et il est conservé lors d'une réinitialisation usine.

Vous pouvez changer librement de Raspberry Pi 5 compatible : l'édition suit la carte. En revanche, une copie du système sur une autre carte n'est associée à aucune édition.

Objectif : vérifier que l'édition active correspond à votre commande.

Étapes :

1. Lisez l'édition affichée à l'étape 2 de l'assistant, ou dans l'état de santé de la sonde.
2. Comparez-la à l'édition indiquée sur l'encart ou l'étiquette du produit.

Vérification : la ressource `GET /api/v1/system` indique l'édition active (§ 6.6.3).

En cas de problème : l'édition dépend du seul certificat d'édition de la carte. Si l'édition affichée n'est pas celle commandée, contactez le support ; si une fonction de l'édition semble limitée, vérifiez aussi que la capacité mémoire du Raspberry Pi 5 respecte les exigences de l'édition (§ 5.3.2), puis contactez le support en indiquant le numéro de série du produit.

> **Information** — Si le certificat d'édition est absent, invalide ou ne correspond pas à la carte (par exemple, système copié sur une autre carte), la sonde démarre en **mode restreint** : aucun contrôle n'est exécuté, mais l'interface et l'API restent accessibles pour consulter l'état, exporter la configuration et produire une archive de diagnostic. L'état de santé indique la cause : champ `editionCertificateStatus` de `GET /api/v1/system`, lisible avec le rôle `viewer` ; le détail figure dans `GET /api/v1/settings/license` (rôle `admin`). Une carte d'origine défectueuse est remplacée selon le § 20.6.

6.6 Vérifier la disponibilité de la sonde

6.6.1 Consulter l'état depuis l'interface d'administration

La page d'accueil de l'interface présente l'état de santé de la sonde : version, édition, durée de fonctionnement, état des moteurs, utilisation des ressources, synchronisation horaire, état des destinations d'export.

Les mêmes informations sont disponibles par l'API, avec la ressource `GET /api/v1/system` (§ 6.6.3).

6.6.2 Vérifier la vivacité avec `/health`

L'adresse `/health` répond sans authentification et indique uniquement si la sonde est en fonctionnement. Elle ne révèle aucune information sensible. Elle convient à une vérification rapide ou à une supervision de premier niveau.

La commande suivante utilise le fichier `sonde-paris-01.pem`, récupéré et vérifié selon le § 6.4.4 (vérification en ligne de commande) :

```bash
# Vérifier la vivacité de la sonde (exemple fictif)
# --cacert : fait confiance au certificat exporté de la sonde, et à lui seul
curl --cacert sonde-paris-01.pem https://192.0.2.50/health
```

Résultat attendu : un code HTTP `200`. La réponse est `{"status": "OK"}`.

> **Avertissement** — L'option `-k` (ou `--insecure`) de `curl` désactive toute vérification du certificat : elle expose vos jetons d'API à une interception. Ne l'utilisez pas, même pour un test ponctuel, en dehors du téléchargement initial du certificat décrit au § 6.4.4.

6.6.3 Créer un jeton d'API et consulter l'état du système

Les appels à l'API, hors `/health` et `GET /api/v1/system/certificate`, exigent un **jeton d'API**. Un jeton d'API est une chaîne aléatoire, associée à un rôle, qui authentifie un outil ou un script auprès de la sonde. La gestion des jetons est décrite au chapitre 9.

Étapes :

1. Dans l'interface d'administration, créez un jeton de rôle `viewer` à usage de vérification (chapitre 9).
2. Copiez-le immédiatement : il n'est affiché qu'une seule fois.
3. Enregistrez-le une seule fois dans un fichier d'en-tête protégé, `$HOME/.sva/entete-auth`, lisible par vous seul, avec la commande ci-dessous (procédure détaillée au § 9.3.3). Le jeton n'apparaît ainsi ni dans l'historique du shell ni dans les arguments des commandes, visibles dans la liste des processus : `curl` lit l'en-tête dans le fichier.

   Sur un poste Linux ou macOS (shell bash ou zsh), la commande suivante rend le répertoire `$HOME/.sva` privé, crée le fichier vide en mode `600`, puis y écrit le jeton, saisi au clavier sans affichage (collez-le, puis appuyez sur Entrée) :

   ```bash
   (
     umask 077
     mkdir -p "$HOME/.sva" && chmod 700 "$HOME/.sva" &&
     rm -f "$HOME/.sva/entete-auth" &&
     : > "$HOME/.sva/entete-auth" && chmod 600 "$HOME/.sva/entete-auth" &&
     IFS= read -rs SONDE_TOKEN &&
     printf 'Authorization: Bearer %s\n' "$SONDE_TOKEN" > "$HOME/.sva/entete-auth"
   )
   ```

   N'activez pas la trace du shell (`set -x`) pendant cette opération : elle afficherait le jeton. Pour remplacer le jeton, recréez simplement le fichier avec la même commande.

```bash
# Consulter l'état du système
# L'en-tête d'authentification est lu dans le fichier protégé créé selon le § 9.3.3
curl --cacert sonde-paris-01.pem \
     -H @"$HOME/.sva/entete-auth" \
     https://192.0.2.50/api/v1/system
```

Résultat attendu : une réponse JSON indiquant notamment la version, l'édition, l'état des modules et des exports. Le détail des champs figure au chapitre 17 (§ 17.9.1).

6.6.4 Contrôler le fonctionnement des services

La sonde repose sur deux composants logiciels principaux : le **moteur principal**, qui planifie et exécute les contrôles et sert l'interface, et le **moteur de scénarios**, qui exécute les scénarios navigateur et les plugins dans un environnement isolé (chapitre 3).

Vérification	Où	Résultat attendu
Moteur principal actif	Page d'état de santé ; `GET /api/v1/system`	État nominal
Moteur de scénarios actif	Idem	État nominal. Hors édition Enterprise, les scénarios navigateur ne sont pas disponibles, mais l'environnement isolé reste toujours actif, quelle que soit l'édition, car il exécute les contrôles ICMP (`fping`) et les plugins. Chromium est présent sur la carte de toutes les éditions, qui partagent le même système, mais il n'est activé que par le certificat d'édition Enterprise (§ 6.5)
Heure synchronisée	Idem	Synchronisation NTP établie
Espace de stockage	Idem	Occupation faible sur une sonde neuve
Alimentation	Idem	Aucune alerte de sous-tension
Mises à jour (composant `updates`)	Idem	`OK` sur une sonde neuve dotée d'au moins une fenêtre de maintenance ; `WARNING` tant qu'aucune fenêtre n'est définie en mode automatique (§ 6.3.3 ; état de santé : chapitre 17, § 17.9.1)

> **Information** — L'accès SSH est désactivé par défaut. Les vérifications courantes ne le nécessitent pas. L'activation temporaire de SSH pour un diagnostic avancé (durée limitée, 4 heures par défaut et 72 heures au maximum, désactivation automatique à l'échéance) est décrite au § 7.9.4.

> **Information** — Si votre sonde est équipée d'un SSD NVMe facultatif (carte M.2 HAT+), celui-ci n'intervient pas dans la mise en service : la sonde démarre et fonctionne sur la microSD, qui reste son support de démarrage. Le SSD s'active **après** la mise en service, par un administrateur ; ses données (historique, artefacts, file d'export, journaux persistants) y sont alors migrées (chapitre 20).

6.7 Premier contrôle de supervision

Objectif : vérifier de bout en bout que la sonde exécute un contrôle, en enregistre le résultat et le restitue.

Prérequis : une cible HTTPS joignable depuis la sonde. L'exemple utilise une application fictive, `https://intranet.example.internal/`. Choisissez une page de votre intranet, accessible sans authentification.

6.7.1 Créer le contrôle depuis l'interface d'administration

Étapes :

1. Ouvrez la gestion des contrôles et créez un contrôle de type **HTTP**.
2. Renseignez :

- identifiant : `intranet-accueil` ;
- nom : « Intranet : page d'accueil » ;
- URL : `https://intranet.example.internal/` ;
- fréquence : 5 minutes ;
- délai maximal (*timeout*) : 10 secondes ;
- vérification : code de réponse égal à 200.

3. Enregistrez puis déclenchez une exécution immédiate.
4. Ouvrez le détail de l'exécution.

Le détail de l'exécution indique l'état du contrôle, sa durée et les phases de la requête. La même opération par l'API est décrite au § 6.7.2.

6.7.2 Créer le contrôle par l'API

Le même contrôle peut être créé par l'API, avec un jeton de rôle `operator` ou `admin` : le jeton `viewer` du § 6.6.3 ne suffit pas. Préparez d'abord ce jeton :

1. Dans l'interface d'administration, ouvrez la gestion des jetons et créez un jeton nommé explicitement (par exemple `mise-en-service`), de rôle `operator`, avec une date d'expiration proche (§ 9.5.2).
2. Copiez sa valeur immédiatement : elle n'est affichée qu'une seule fois.
3. Remplacez le jeton `viewer` par ce jeton dans le fichier d'en-tête protégé `$HOME/.sva/entete-auth`, en exécutant de nouveau la commande du § 6.6.3 (fichier recréé en mode `600`, jeton saisi sans affichage).

L'exemple suppose que le fichier `intranet-accueil.json` contient la définition suivante :

```json
{
  "id": "intranet-accueil",
  "name": "Intranet : page d'accueil",
  "type": "http",
  "schedule": { "interval": "5m" },
  "timeout": "10s",
  "http": {
    "method": "GET",
    "url": "https://intranet.example.internal/",
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 }
    ]
  }
}
```

```bash
# 1. Créer le contrôle
curl --cacert sonde-paris-01.pem \
     -H @"$HOME/.sva/entete-auth" \
     -H "Content-Type: application/json" \
     --data @intranet-accueil.json \
     https://192.0.2.50/api/v1/checks

# 2. Déclencher une exécution immédiate (réponse 202 Accepted + en-tête Location)
curl --cacert sonde-paris-01.pem -i -X POST \
     -H @"$HOME/.sva/entete-auth" \
     https://192.0.2.50/api/v1/checks/intranet-accueil/runs

# 3. Consulter le dernier état du contrôle
curl --cacert sonde-paris-01.pem \
     -H @"$HOME/.sva/entete-auth" \
     https://192.0.2.50/api/v1/checks/intranet-accueil/status
```

Résultat attendu :

- la création renvoie le contrôle enregistré ;
- le déclenchement renvoie `202 Accepted` et l'adresse de l'exécution dans l'en-tête `Location` ;
- après quelques secondes, l'état vaut `OK`.

Le modèle complet des contrôles est décrit aux chapitres 8 et 10.

> **Information** — L'état d'un contrôle (`GET /api/v1/checks/{id}/status` , commande 3 de l'exemple) se lit avec un jeton de rôle `metrics` ou supérieur : `metrics` est le rôle minimal indiqué pour cette ressource (§ 9.6, chapitre 21). Chaque jeton possède un rôle unique (§ 9.4.1) ; le jeton `operator` utilisé ici convient donc pour les trois commandes, et un jeton `metrics` suffit pour la seule commande 3, par exemple pour Nagios ou Centreon.

En cas de problème :

État obtenu	Cause probable	Action
`CRITICAL` avec `error.type` = `dns_error`	Nom non résolu par les serveurs DNS de la sonde	Vérifier la configuration DNS (§ 7.3)
`CRITICAL` avec `connection_refused` ou `timeout`	Flux bloqué par un pare-feu, ou proxy nécessaire	Vérifier les flux (§ 7.6) et le proxy (§ 7.5)
`CRITICAL` avec `tls_error`	Certificat de la cible émis par une AC interne inconnue de la sonde	Importer l'AC interne (§ 7.7.3)
`CRITICAL` avec `http_status`	Code de réponse différent de 200 (redirection vers une page d'authentification, par exemple)	Choisir une page publique ou adapter l'assertion

6.8 Vérifier la remontée des résultats

Les résultats sont conservés **localement** par la sonde, qui fonctionne de manière autonome : aucune plateforme externe n'est nécessaire pour les consulter.

Vérification	Méthode	Résultat attendu
Historique local	Interface : historique du contrôle ; API : `GET /api/v1/runs?checkId=intranet-accueil`	Les exécutions planifiées apparaissent toutes les 5 minutes

Vérification	Méthode	Résultat attendu
Destination d'export (si configurée)	Interface : état des destinations ; API : `POST /api/v1/exporters/{id}/test`	Envoi de test réussi
Métriques Prometheus (si utilisé)	Collecte de `/metrics` par votre serveur Prometheus	La série de l'état du contrôle est présente (chapitre 18)

La configuration des destinations d'export (Prometheus, Grafana, Datadog, Dynatrace, Nagios/Centreon) est décrite au chapitre 18 ; l'interprétation des résultats, au chapitre 17.

> ✓ **Conseil** — Une fois le premier contrôle validé, révoquez (supprimez) les jetons `viewer` et `operator` créés pour ces vérifications, ou créez-les dès le départ avec une date d'expiration (`expiresAt`) (chapitre 9, § 9.5).

6.9 Checklist de fin de mise en service

Point de contrôle	Référence	Vérifié
Premier démarrage terminé sans interruption	§ 6.2	[ ]
Provisionnement appliqué sans erreur (méthode A) ou assistant terminé (méthode B)	§ 6.3, § 6.4	[ ]
Empreinte du certificat vérifiée et consignée	§ 6.4.4	[ ]
Compte administrateur personnel créé ; mot de passe stocké dans le coffre-fort de l'organisation	§ 6.3.4, § 6.4.5	[ ]
Code de récupération du compte administrateur enregistré dans le coffre-fort de l'organisation ; encart ou étiquette portant le code d'installation conservé	§ 6.4.6 ; chapitre 20, § 20.10.2	[ ]
Authentification 802.1X établie, si le port est contrôlé	§ 6.3.3 ; chapitre 7	[ ]
Adresse IP définitive confirmée (réservation DHCP ou adresse statique)	§ 7.2	[ ]
Nom d'hôte et site renseignés	§ 6.3.3	[ ]
Heure synchronisée par NTP	§ 7.4	[ ]
Édition affichée conforme à la commande	§ 6.5	[ ]
Moteurs en fonctionnement, aucune alerte de santé	§ 6.6.4	[ ]
Premier contrôle exécuté avec l'état `OK`	§ 6.7	[ ]
Résultats consultables localement	§ 6.8	[ ]
Jetons `viewer` et `operator` créés pour les vérifications révoqués, ou dotés d'une date d'expiration (`expiresAt`)	§ 6.8 ; chapitre 9, § 9.5	[ ]
Destinations d'export configurées et testées (si prévu)	Chapitre 18	[ ]
Fichier de provisionnement et, le cas échéant, fichiers 802.1X (PKCS#12, certificat de l'AC) supprimés de la carte et du poste de préparation	§ 6.3	[ ]
SSH toujours désactivé (sauf besoin justifié), ou verrouillé si votre politique de sécurité l'exige	§ 6.3.3, § 7.9.4	[ ]
Politique de mises à jour définie : mode de chaque canal et au moins une fenêtre de maintenance, ou mode manuel assumé avec une procédure d'activation par un administrateur ; état de santé `updates` vérifié	§ 6.3.3, § 6.4.5 ; chapitre 20	[ ]
SSD NVMe éventuel activé une fois la mise en service terminée	Chapitre 20	[ ]
Fiche de mise en service complétée (chapitre 21)	Chapitre 21	[ ]

La sonde est opérationnelle. Le chapitre 7 détaille la configuration réseau et les mesures de sécurité à appliquer pour une exploitation durable.

Chapitre 7 — Configuration réseau et sécurité

La sonde s'installe dans le réseau qui correspond au point d'observation choisi : réseau de votre organisation, site distant, ou réseau d'un client, d'un partenaire ou d'un sous-traitant (chapitre 3, § 3.2.1). Elle accède à des applications métier, parfois avec des comptes de test, et transmet des données vers vos outils de supervision. Sa configuration réseau et sa sécurisation conditionnent donc à la fois la **pertinence des mesures** et la **sécurité du système d'information** qui l'accueille.

Ce chapitre s'adresse aux administrateurs réseau et système ainsi qu'aux responsables de la sécurité des systèmes d'information (RSSI). Il complète la mise en service (chapitre 6) et précède l'administration courante (chapitre 9).

> **Information** — Les adresses, noms de domaine et comptes utilisés dans ce chapitre sont fictifs : réseau `192.0.2.0/24`, domaine `example.internal`.

7.1 Principes de sécurité de la sonde

La conception de la sonde repose sur les principes suivants, qui guident l'ensemble des recommandations de ce chapitre.

Principe	Mise en œuvre
Aucun secret par défaut	Pas de mot de passe par défaut ; clés et certificat générés sur chaque sonde au premier démarrage (§ 6.2)
Connexions sortantes privilégiées	La sonde initie elle-même ses échanges avec les plateformes externes. Les flux entrants se limitent à l'administration et à l'API en HTTPS (443/TCP), à la redirection HTTP vers HTTPS (80/TCP, désactivable), à l'annonce mDNS (5353/UDP, désactivable), à SSH (22/TCP) s'il est activé, et à la collecte par Prometheus ou l'interrogation par Nagios et Centreon, en HTTPS (§ 7.6.1)
Chiffrement des échanges	Interface d'administration et API exclusivement en HTTPS ; exports vers les plateformes en HTTPS. L'export OTLP en HTTP non chiffré reste admis vers un collecteur ou un agent interne (§ 7.6.1)
Moindre privilège	Rôles distincts pour les utilisateurs et les jetons ; comptes de test dédiés et limités ; exécution des scénarios et plugins dans un environnement isolé, sans accès aux secrets non déclarés
Autonomie	La sonde n'exige aucun accès à Internet pour fonctionner ; chaque flux externe est facultatif et explicite
Traçabilité	Journal d'audit des opérations d'administration

> **Avertissement** — La sonde n'est pas conçue pour être exposée sur Internet. N'ouvrez jamais l'accès à son interface d'administration ou à son API depuis l'extérieur du réseau où elle est installée, même temporairement. Pour une administration à distance, utilisez un moyen d'accès sécurisé à ce réseau (VPN d'administration, bastion), celui de votre organisation ou celui convenu avec l'exploitant du réseau d'accueil.

7.2 Adressage IP

La sonde utilise une seule interface réseau, l'interface Ethernet. Elle fonctionne en IPv4, toujours actif, et peut fonctionner en double pile : l'IPv6 est activable et désactivé par défaut (§ 7.2.5). L'accès à un port protégé par 802.1X est décrit au § 7.2.6.

7.2.1 DHCP et réservation

Par défaut, la sonde obtient son adresse par **DHCP** (*Dynamic Host Configuration Protocol*), le protocole qui attribue automatiquement une adresse IP, une passerelle et des serveurs DNS aux équipements d'un réseau.

L'adresse d'une sonde doit cependant rester **stable**, car vos outils l'interrogent : serveur Prometheus, Nagios ou Centreon, postes d'administration, règles de pare-feu. Deux solutions sont possibles :

Solution	Avantages	Inconvénients
Réservation DHCP (adresse fixe associée à l'adresse MAC)	Configuration centralisée sur le serveur DHCP ; aucune intervention sur la sonde ; changement d'adressage maîtrisé par l'équipe réseau	Dépend de la disponibilité du serveur DHCP au démarrage
Adresse statique configurée sur la sonde	Indépendante du DHCP	Configuration à maintenir sur chaque sonde ; risque de conflit d'adresse si la plage n'est pas réservée

> ✓ **Conseil** — Privilégiez la réservation DHCP lorsque votre organisation en a l'usage. Réservez l'adresse statique aux réseaux sans DHCP, comme certains réseaux de serveurs.

7.2.2 Adresse statique

Une configuration statique comprend :

- l'adresse IPv4 et la longueur de préfixe (exemple : `192.0.2.50/24`) ;
- la passerelle par défaut (exemple : `192.0.2.1`) ;
- un ou plusieurs serveurs DNS (deux recommandés) ;
- éventuellement un ou plusieurs domaines de recherche DNS (propriété `searchDomains` de la section `network`).

Elle peut être définie :

- dès le premier démarrage, par le fichier de provisionnement (§ 6.3) ;
- lors de la mise en service, par l'assistant (§ 6.4.5) ;
- à tout moment ensuite, par l'interface d'administration ou par l'API (§ 7.2.4).

7.2.3 Mécanisme de confirmation des changements réseau

Une erreur de configuration réseau (adresse incorrecte, passerelle erronée) peut rendre une sonde injoignable, et imposer une intervention physique. Pour éviter cette situation, la sonde applique tout changement de configuration réseau selon un principe de **confirmation obligatoire** :

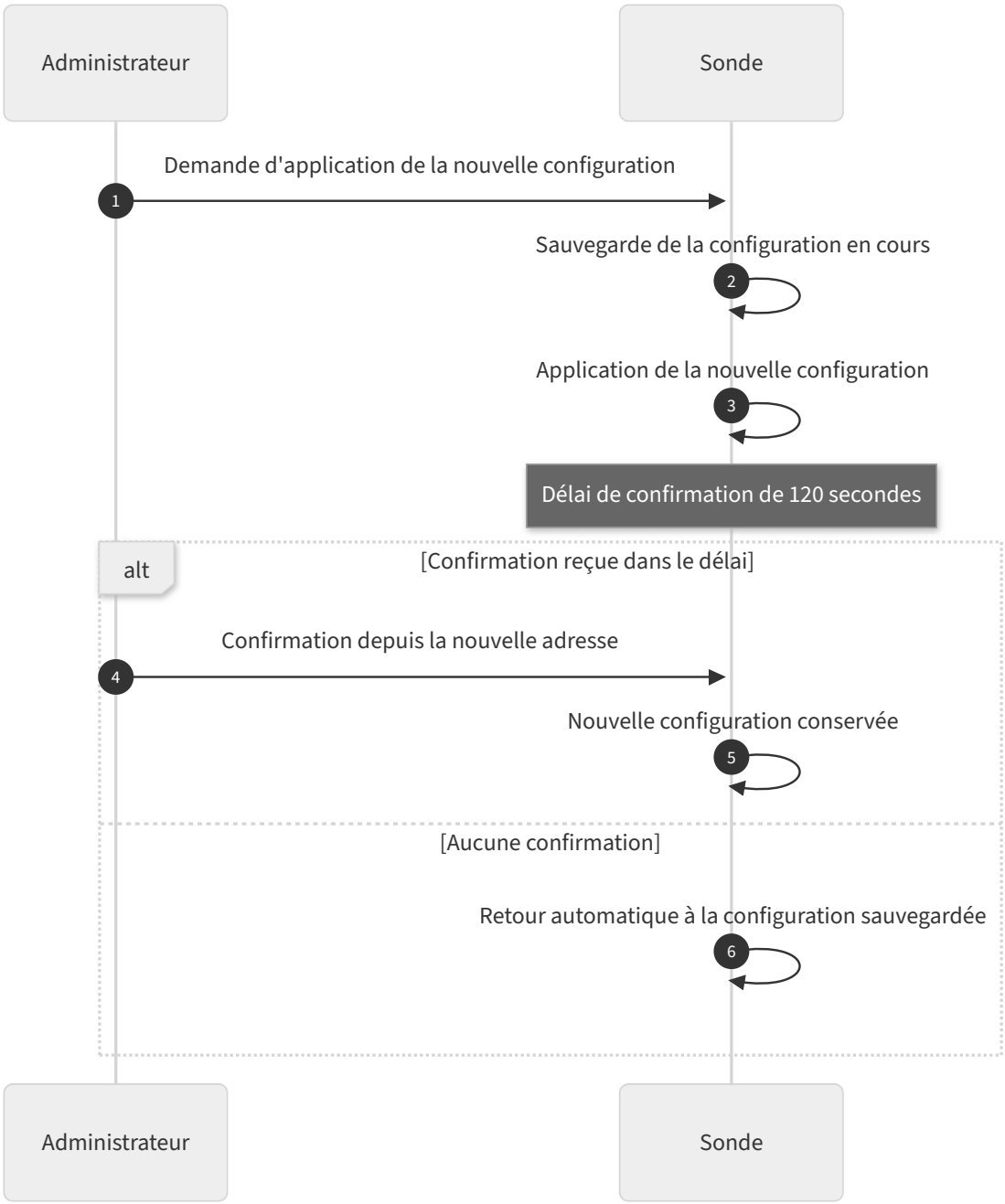


Schéma 7-1 — Confirmation obligatoire d'un changement réseau

Conséquences pratiques :

1. Après l'application, la sonde peut changer d'adresse : reconnectez-vous à la **nouvelle** adresse.
2. Confirmez dans les **120 secondes**.

3. Si vous ne parvenez pas à joindre la nouvelle adresse, attendez : au terme du délai, la sonde revient d'elle-même à l'ancienne configuration.

L'application (`POST /api/v1/settings/network/apply`) renvoie un identifiant de changement. La confirmation s'effectue par l'appel `POST /api/v1/settings/network/confirm`, émis **depuis la nouvelle adresse** de la sonde, ce qui prouve qu'elle reste joignable. L'interface d'administration émet cette confirmation lorsque, dans le délai, vous vous êtes reconnecté à la nouvelle adresse et authentifié de nouveau (paragraphe « Confirmation depuis un navigateur » ci-dessous).

Certificat pendant et après la confirmation. Pendant la fenêtre de 120 secondes, la sonde présente encore l'**ancien certificat**, celui dont vous avez déjà vérifié l'empreinte. Vous confirmez donc à la nouvelle adresse en reconnaissant la même empreinte (interface d'administration), ou en indiquant à curl l'ancien certificat et l'ancien nom (API, § 7.2.4). Si le changement entraîne la régénération du certificat autosigné (encadré du § 7.2.4), la réponse à la confirmation contient le **nouveau certificat** : champ `certificate` (format PEM), champ `fingerprint` (empreinte SHA-256) et champ `activatesAt` (instant de la mise en service, 30 secondes après la confirmation). Transmis par la connexion déjà vérifiée, ce certificat est une source de confiance : aucune vérification hors réseau n'est nécessaire dans le délai. L'interface d'administration affiche la nouvelle empreinte avant la bascule.

Confirmation depuis un navigateur. La session d'administration est liée à l'origine, c'est-à-dire à l'adresse ou au nom saisi dans le navigateur : son cookie et le jeton de protection contre la falsification de requêtes (CSRF, *Cross-Site Request Forgery*) ne sont pas valables à la nouvelle adresse. Après avoir reconnu l'empreinte à la nouvelle adresse, **authentifiez-vous de nouveau** ; la confirmation n'est émise qu'ensuite. Tenez compte de ce temps de reconnexion dans le délai de 120 secondes.

Recours lorsque le parcours ne se déroule pas comme prévu.

Situation	Conduite à tenir
La politique du navigateur interdit l'exception de sécurité à la nouvelle adresse (navigateur géré par l'organisation, par exemple)	Laissez la sonde revenir à l'ancienne configuration (120 secondes), puis effectuez le changement par l'API, avec `curl` (§ 7.2.4). Pour éviter ce cas, préférez une réservation DHCP (§ 7.2.1), qui ne modifie pas la configuration de la sonde, ou installez au préalable un certificat émis par l'AC de votre entreprise (§ 7.7.2)
La réponse de confirmation est perdue (coupure, fermeture de l'onglet, échec de l'écriture du fichier `confirmation.json`)	Récupérez le certificat par `GET /api/v1/system/certificate` (sans authentification, § 6.4.4), puis vérifiez son empreinte par un canal indépendant du réseau : de préférence la console de l'écran HDMI. Le fichier `setup-code.txt` de la partition `SVA-BOOT` n'est réécrit qu'au démarrage : il peut encore porter l'empreinte de l'ancien certificat tant que la sonde n'a pas redémarré (chapitre 6, § 6.2). Ne transmettez aucun jeton et ne saisissez aucun identifiant tant que cette vérification n'est pas faite
Vous ne savez pas si la confirmation a été enregistrée	Attendez la fin du délai de 120 secondes : la sonde est alors joignable à la nouvelle adresse si la confirmation a été enregistrée, à l'ancienne dans le cas contraire. Le journal d'audit l'indique
Aucune confirmation dans le délai	La sonde revient automatiquement à la configuration précédente au bout de 120 secondes : c'est la sécurité du mécanisme. Recommencez l'opération

✓ **Conseil** — Ne changez l'adresse ou le nom d'hôte d'une sonde qu'en cas de nécessité. Pour stabiliser une adresse, préférez une réservation DHCP (§ 7.2.1), qui ne modifie pas la configuration de la sonde.

7.2.4 Modifier la configuration réseau

Objectif : modifier l'adressage d'une sonde en service sans risque de perte d'accès.

Prérequis :

- compte ou jeton de rôle `admin` ;

- nouveaux paramètres validés avec l'équipe réseau ;
- règles de pare-feu mises à jour pour la nouvelle adresse (accès d'administration, collecte Prometheus, Nagios) ;
- poste d'administration capable de joindre la nouvelle adresse.

Étapes (interface d'administration) :

1. Ouvrez les paramètres réseau de l'interface d'administration.
2. Saisissez les nouveaux paramètres.
3. Appliquez.
4. Reconnectez-vous à la nouvelle adresse dans les 120 secondes : l'interface confirme alors la configuration.

Étapes (API) : la section `network` des paramètres se consulte et se modifie par `GET` et `PUT /api/v1/settings/network`, puis s'applique par `POST /api/v1/settings/network/apply` et se confirme par `POST /api/v1/settings/network/confirm` (§ 7.2.3). Le format de la section `network` est le suivant.

JSON

```
{
  "hostname": "sonde-paris-01",
  "mode": "static",
  "address": "192.0.2.50/24",
  "gateway": "192.0.2.1",
  "dns": ["192.0.2.10", "192.0.2.11"]
}
```

BASH

```
# Préparation du poste selon le § 9.3.3 : fichier d'en-tête $HOME/.sva/entete-auth
# (jeton de rôle admin), variable CA (certificat vérifié de la sonde) et fonction
# sapi. SONDE désigne ici l'ancienne adresse de la sonde, présente dans son
# certificat (exemple fictif).
SONDE=https://192.0.2.50

# 1. Lire la configuration courante et son ETag (en-têtes affichés par -i)
sapi -i "$SONDE/api/v1/settings/network"

# 2. Enregistrer la nouvelle configuration (If-Match : ETag relevé à l'étape 1)
sapi -X PUT -H 'Content-Type: application/json' \
     -H 'If-Match: "<etag>"' \
     --data @network.json \
     "$SONDE/api/v1/settings/network"

# 3. Appliquer : la réponse contient l'identifiant du changement, à confirmer
#    dans les 120 secondes par POST /api/v1/settings/network/confirm,
#    appelé depuis la nouvelle adresse de la sonde (§ 7.2.3)
sapi -X POST "$SONDE/api/v1/settings/network/apply"

# 4. Confirmer depuis la nouvelle adresse (exemple fictif : 192.0.2.60), dans les
#    120 secondes. La sonde présente encore l'ancien certificat : curl le vérifie
#    avec l'ancien fichier de confiance ($CA) et l'ancien nom (ici l'ancienne
#    adresse, 192.0.2.50), et se connecte à la nouvelle adresse grâce à --connect-to.
#    <id> : identifiant renvoyé à l'étape 3
sapi --connect-to 192.0.2.50:443:192.0.2.60:443 \
     -X POST -H 'Content-Type: application/json' \
     --data '{"id":"<id>"}' \
     -o confirmation.json \
     "$SONDE/api/v1/settings/network/confirm"
```

Si le certificat autosigné est régénéré, la réponse de l'étape 4 (fichier `confirmation.json`) contient les champs `certificate`, `fingerprint` et `activatesAt` (§ 7.2.3). Enregistrez le contenu du champ `certificate` comme **nouveau fichier de confiance** (par exemple `sonde-paris-01-nouveau.pem`, avec l'outil standard `jq` : `jq -r`

`.certificate confirmation.json > sonde-paris-01-nouveau.pem`) et relevez l'empreinte `fingerprint`. À partir de l'instant `activatesAt`, utilisez ce nouveau fichier dans vos appels (variable `CA` du § 9.3.3, option `--cacert`) et fournissez-le à vos outils (Prometheus, Nagios, Centreon).

Si la commande de l'étape 4 n'a pas produit de fichier `confirmation.json` exploitable, appliquez les recours du § 7.2.3 : récupération du certificat par `GET /api/v1/system/certificate`, puis vérification de son empreinte sur la console HDMI ou dans `setup-code.txt`, avant tout nouvel appel muni du fichier d'en-tête.

> **Information** — Dans l'étape 4, `--connect-to` dirige la connexion vers la nouvelle adresse tout en conservant l'ancien nom pour la vérification du certificat, qui contient encore ce nom. Le nom utilisé dans l'URL doit figurer dans l'ancien certificat (nom d'hôte, nom mDNS ou ancienne adresse IPv4, comme dans l'exemple).

> **Attention** — `PUT /api/v1/settings/network` **remplace la section entière** : un champ absent du corps reprend sa valeur par défaut ou disparaît. Partez donc toujours du résultat du `GET` de l'étape 1, et conservez dans `network.json` les champs que vous ne modifiez pas, notamment `mdns`, `searchDomains` et `ipv6`. Les exemples ci-dessus ne montrent que les champs de l'adressage IPv4.

> **Information** — Les serveurs NTP ne font pas partie de la section `network` des paramètres : le champ `network.ntp` du fichier de provisionnement (§ 6.3.3) correspond à la section `time` des paramètres (§ 7.4).

L'en-tête `If-Match` garantit que la configuration n'a pas été modifiée par un autre administrateur entre la lecture et l'écriture : en cas de conflit, la sonde répond `412 Precondition Failed`.

Résultat attendu : la sonde est joignable à sa nouvelle adresse ; la configuration est confirmée.

Vérification : page d'état de santé ; exécution correcte des contrôles ; collecte Prometheus ou interrogation Nagios rétablie.

En cas de problème : sans confirmation, la sonde revient à l'ancienne configuration au bout de 120 secondes. Si elle reste injoignable au-delà, voir chapitre 19.

> **Attention** — Le certificat autosigné de la sonde contient son nom d'hôte, son nom mDNS, son adresse IPv4 et, si l'IPv6 est activé en mode statique, son adresse IPv6. Si l'un de ces éléments change et qu'aucun certificat importé n'est en place, la sonde régénère automatiquement son certificat autosigné ; le nouveau certificat apparaît sur la console HDMI et dans le journal d'audit. Pendant la fenêtre de confirmation, la sonde présente encore l'ancien certificat ; le nouveau est transmis dans la réponse de la confirmation, par la connexion déjà vérifiée, puis mis en service 30 secondes plus tard (§ 7.2.3). Son empreinte change : les outils auxquels vous avez fourni l'ancien certificat (curl, Prometheus) refusent alors la connexion jusqu'à ce que vous leur fournissiez le nouveau, enregistré à partir de la réponse de la confirmation (étape 4 ci-dessus). Utilisez un nom DNS stable plutôt qu'une adresse IP dans vos configurations, et de préférence un certificat de votre AC interne (§ 7.7.2).

7.2.5 IPv6

La sonde fonctionne en **double pile** : l'IPv4 est toujours actif et obligatoire, l'IPv6 est facultatif et **désactivé par défaut**. Une fois activé, il permet d'administrer la sonde et de superviser des cibles en IPv6, en complément de l'IPv4.

Paramètres. L'IPv6 se règle dans la section `network`, par l'objet `ipv6` :

Champ	Description	Exemple fictif
`ipv6.enabled`	Active l'IPv6. Valeur par défaut : `false`	`true`
`ipv6.mode`	`slaac` : adresse obtenue automatiquement à partir des annonces du routeur (*StateLess Address AutoConfiguration*) ; `static` : adresse fixe	`static`
`ipv6.address`	Adresse IPv6 et longueur de préfixe, en notation CIDR. Obligatoire en mode `static`	`2001:db8:10::50/64`
`ipv6.gateway`	Passerelle IPv6 par défaut. Obligatoire en mode `static`	`2001:db8:10::1`

Les serveurs DNS de la sonde (propriété `dns`) peuvent indifféremment être des adresses IPv4 ou IPv6.

Exemple de section `network` avec IPv4 et IPv6 statiques (valeurs fictives) :

```json
{
  "hostname": "sonde-paris-01",
  "mode": "static",
  "address": "192.0.2.50/24",
  "gateway": "192.0.2.1",
  "dns": ["192.0.2.10", "2001:db8:10::53"],
  "ipv6": {
    "enabled": true,
    "mode": "static",
    "address": "2001:db8:10::50/64",
    "gateway": "2001:db8:10::1"
  }
}
```

En mode `slaac`, l'objet se réduit à `{"enabled": true, "mode": "slaac"}`.

Effets de l'activation.

Domaine	Comportement
Contrôles	Le paramètre commun `ipFamily` de chaque contrôle choisit la famille d'adresses : `auto` (par défaut) utilise l'IPv6 si celui-ci est activé et si la cible possède une adresse AAAA, l'IPv4 sinon ; `ipv4` et `ipv6` imposent une famille (chapitre 8). Les contrôles ICMP utilisent alors ICMPv6 (outil `fping`)
Pare-feu local	Les règles du pare-feu local (§ 7.6.3) s'appliquent de façon identique en IPv6
Certificat autosigné	S'il est en place, il inclut l'adresse IPv6 statique de la sonde
Flux réseau	Le flux ICMPv6 de découverte des voisins doit être autorisé sur le réseau local de la sonde (§ 7.6.1)

Procédure.

Objectif : activer l'IPv6 sur une sonde en service.

Prérequis :

- compte ou jeton de rôle `admin` ;
- réseau de la sonde routé en IPv6, avec annonces de routeur en mode `slaac`, ou adresse et passerelle IPv6 attribuées par l'équipe réseau en mode `static` ;
- règles de pare-feu réseau mises à jour pour l'IPv6 : accès d'administration, collecte Prometheus, Nagios, flux des contrôles, ICMPv6 de découverte des voisins.

Étapes :

1. Ajoutez l'objet `ipv6` à la section `network`, dans l'interface d'administration ou par l'API (§ 7.2.4). L'IPv6 peut aussi être activé dès la mise en service, par le fichier de provisionnement (§ 6.3.3).
2. Appliquez la configuration, puis confirmez-la dans les 120 secondes (§ 7.2.3).
3. Pour chaque contrôle dont la cible doit être jointe dans une famille précise, réglez `ipFamily` (chapitre 8).

Résultat attendu : la sonde dispose d'une adresse IPv6 et reste joignable en IPv4.

Vérification : la configuration effective, lue par `GET /api/v1/settings/network` ou sur la page d'état de santé, présente l'adresse IPv6 ; un contrôle de test réglé sur `"ipFamily": "ipv6"` vers une cible IPv6 connue réussit.

Erreurs possibles :

Symptôme	Cause probable	Action
Aucune adresse IPv6 en mode `slaac`	Pas d'annonce de routeur sur le réseau de la sonde ; ICMPv6 filtré	Vérifier la configuration du routeur avec l'équipe réseau ; autoriser ICMPv6 de découverte des voisins
Contrôles en échec après l'activation, alors qu'ils réussissaient en IPv4	Les cibles ont une adresse AAAA, mais le chemin IPv6 est filtré ou non routé. La règle `auto` ne prévoit pas de repli sur l'IPv4	Corriger le routage ou le filtrage IPv6, ou fixer `"ipFamily": "ipv4"` sur les contrôles concernés
Interface d'administration inaccessible en IPv6	Pare-feu réseau ; liste des réseaux autorisés de la sonde (§ 7.6.3)	Ouvrir l'accès HTTPS en IPv6 depuis le réseau d'administration ; ajuster la liste des réseaux autorisés

> **Attention** — Une liste de réseaux autorisés (§ 7.6.3) qui ne contient que des préfixes IPv4 ne laisse pas passer l'administration en IPv6. La liste accepte des préfixes IPv4 et IPv6 en notation CIDR (par exemple `2001:db8:10::/64`) : ajoutez-y les préfixes IPv6 de vos postes d'administration.

7.2.6 Authentification 802.1X

Le contrôle d'accès **802.1X** impose à un équipement de s'authentifier auprès du commutateur avant d'obtenir l'accès au réseau. Le commutateur, appelé « authentificateur », transmet l'authentification à un **serveur RADIUS**, qui l'accepte ou la refuse. Tant que l'authentification n'a pas réussi, le port ne laisse passer que les échanges d'authentification (trames EAPOL, *EAP over LAN*).

La sonde prend en charge 802.1X sur sa prise Ethernet, avec la méthode **EAP-TLS** : elle s'authentifie par un **certificat client**, et vérifie elle-même le certificat du serveur RADIUS. Le client 802.1X de la sonde est `wpa_supplicant`, en mode filaire.

Élément	Prise en charge
EAP-TLS (authentification par certificat)	Oui
Exception par adresse MAC (MAB, *MAC Authentication Bypass*)	Oui, configurée sur le commutateur : la sonde n'a rien à paramétrer
PEAP (identifiant et mot de passe)	Non
Étiquetage VLAN (IEEE 802.1Q) sur la prise de la sonde	Non : la sonde se raccorde à un port d'accès

Paramètres. La section `dot1x` des paramètres contient :

Champ	Description
`enabled`	Active l'authentification 802.1X sur la prise Ethernet
`identity`	Identité présentée au serveur RADIUS
`caRef`	Identifiant (`id`) de l'autorité dans la section `trust` : celle qui a émis le certificat du serveur RADIUS (§ 7.7.3)

Exemple fictif :

```json
{
  "enabled": true,
  "identity": "sonde-paris-01.example.internal",
  "caRef": "ac-radius-example"
}
```

L'**identifiant client** (clé privée et certificat) ne figure pas dans cette section. Il s'obtient de deux manières :

Méthode	Appel	Remarque
Génération de la clé et de la demande de certificat (CSR) sur la sonde	`POST /api/v1/system/dot1x/csr`	**Recommandée** : la clé privée ne quitte jamais la sonde
Import d'un fichier PKCS#12 (clé et certificat)	`PUT /api/v1/system/dot1x/credential`	Écriture seule : l'identifiant importé ne peut pas être relu

Ces opérations sont réservées au rôle `admin` et enregistrées dans le journal d'audit.

> **Information** — Les trames EAPOL circulent uniquement entre la sonde et le commutateur, au niveau de la liaison Ethernet : elles ne nécessitent aucune ouverture de flux IP. Les échanges RADIUS ont lieu entre le commutateur et le serveur RADIUS, sans passer par la sonde.

Choisir une voie de mise en œuvre. Un port contrôlé par 802.1X n'offre aucun accès au réseau avant l'authentification : la sonde doit donc disposer de son identifiant avant de pouvoir être jointe. Deux voies sont possibles.

Voie	Principe	À privilégier lorsque
PKCS#12 — Provisionnement	L'identifiant (fichier PKCS#12 et son mot de passe) et le certificat de l'AC du serveur RADIUS sont déposés sur la microSD, avec la section `dot1x` du fichier de provisionnement. La sonde s'authentifie dès son premier démarrage, avant tout accès au réseau	Aucun port non contrôlé ni exception MAB n'est disponible, même temporairement
CSR — Activation après la mise en service	La sonde est mise en service sur un port non contrôlé ou autorisé par MAB, puis 802.1X est activé par l'interface ou l'API, avec une clé générée sur la sonde	Dans les autres cas : c'est la voie recommandée, car la clé privée ne quitte jamais la sonde

Procédure — voie PKCS#12 (provisionnement).

Objectif : obtenir l'accès à un port contrôlé par 802.1X dès le premier démarrage.

Prérequis : prérequis du § 6.3.2 ; fichier PKCS#12 contenant la clé privée et le certificat client de la sonde, émis par votre IGC, et son mot de passe ; certificat de l'AC du serveur RADIUS au format PEM ; identité attendue par le serveur RADIUS.

Étapes :

1. Copiez le fichier PKCS#12 et le certificat de l'AC à la racine de la partition `SVA-BOOT` de la microSD.

2. Ajoutez au fichier `provision.yaml` la section `dot1x` (champs détaillés au § 6.3.3) :

```yaml
# Section 802.1X du fichier de provisionnement — exemple fictif
dot1x:
  identity: sonde-paris-01.example.internal
  pkcs12File: sonde-paris-01.p12      # fichier déposé à la racine de la partition SVA-BOOT
  caFile: radius-ca.pem               # AC du serveur RADIUS, format PEM
  pkcs12Password: "<mot de passe du fichier PKCS#12>"   # seul secret admis dans ce fichier
```

3. Terminez la préparation de la microSD (§ 6.3.5), installez la sonde et raccordez-la au port contrôlé (chapitre 5).

Résultat attendu : au premier démarrage, la sonde s'authentifie, puis obtient sa configuration réseau. Le fichier PKCS#12, le certificat de l'AC et le mot de passe sont supprimés de la microSD après traitement, comme `provision.yaml`.

Vérification : `provision-result.txt` (§ 6.3.7) ; port autorisé sur le commutateur ; sonde joignable à son adresse.

> **Avertissement** — La suppression du fichier PKCS#12 par la sonde ne vaut pas effacement physique de la mémoire flash. Préférez la voie CSR ; si la voie PKCS#12 est indispensable, appliquez les précautions détaillées au chapitre 6 (§ 6.3.5), puis remplacez le certificat dès que possible selon les étapes 3 à 5 de la voie CSR.

> **Avertissement** — Le mot de passe du fichier PKCS#12 est le **seul secret admis** dans le fichier de provisionnement. Entre la préparation de la carte et le premier démarrage, la microSD contient la clé privée de la sonde et ce mot de passe : préparez les fichiers sur un poste de confiance, supprimez-en toute copie, et ne laissez pas la carte sans surveillance (§ 6.3.3).

> **Attention** — L'authentification EAP-TLS vérifie la validité des certificats, ce qui suppose une heure correcte. Or, au premier démarrage sur un port 802.1X, le serveur NTP n'est joignable qu'après l'authentification. Sans pile pour son horloge temps réel, la sonde part de la dernière heure enregistrée ou, à défaut, de la date de construction de l'image. Pour un déploiement 802.1X, équipez le Raspberry Pi 5 de la pile officielle de l'horloge temps réel (§ 7.4). Une pile neuve ne donne l'heure juste qu'après un premier réglage de l'horloge, c'est-à-dire après un démarrage au cours duquel la sonde a accès à un serveur NTP (par exemple sur un port non contrôlé ou autorisé par MAB).

Pré-régler l'horloge avant un premier démarrage sur un port 802.1X. Les chemins suivants restent sous votre contrôle :

Chemin	Principe	Condition
Premier démarrage sur un port non contrôlé	Avec la pile officielle de l'horloge temps réel installée, démarrez une première fois la sonde sur un port sans 802.1X, depuis lequel le flux NTP (UDP 123) est autorisé ; la sonde règle son horloge, qui conserve ensuite l'heure hors tension. Arrêtez-la proprement, puis raccordez-la au port contrôlé	Un port non contrôlé, même temporaire, sur un réseau qui donne accès à un serveur NTP
Exception MAB temporaire	L'équipe réseau autorise provisoirement le port par l'adresse MAC de la sonde (MAB), le temps d'un premier démarrage avec accès NTP, puis retire l'exception	Une politique de sécurité qui admet une exception MAB limitée dans le temps

Si aucun de ces chemins n'est possible, la réussite de l'authentification EAP-TLS au premier démarrage **ne peut pas être garantie** : elle dépend de l'écart entre l'heure de départ de la sonde (dernière heure enregistrée ou date de construction de l'image) et les périodes de validité des certificats, notamment celle du certificat du serveur RADIUS,

que la sonde vérifie. La sonde ne désactive jamais la vérification des périodes de validité TLS pour contourner ce cas. En cas d'échec, la console de l'écran HDMI affiche l'état de l'authentification et permet de distinguer une heure incorrecte des autres causes.

Procédure — voie CSR (activation après la mise en service).

Objectif : basculer une sonde en service vers un port contrôlé par 802.1X, avec une clé générée sur la sonde.

Prérequis :

- sonde mise en service sur un port non contrôlé ou autorisé par MAB (chapitre 6) ;
- compte ou jeton de rôle `admin` ;
- certificat de l'AC du serveur RADIUS au format PEM ;
- accès à votre IGC pour faire signer une demande de certificat client ;
- heure de la sonde synchronisée (§ 7.4) : la validité des certificats en dépend.

Étapes :

1. Importez le certificat de l'AC du serveur RADIUS dans le magasin de confiance de la sonde, section `trust` (§ 7.7.3), et notez l'identifiant (`id`) sous lequel il est enregistré.
2. Renseignez la section `dot1x` **sans l'activer** : `identity` et `caRef`, avec `enabled` à `false`, dans l'interface d'administration ou par `PUT /api/v1/settings/dot1x`, avec l'en-tête `If-Match` (§ 7.2.4). L'identité doit être enregistrée avant la demande de certificat, dont elle constitue le sujet.

   ```json
   {
     "enabled": false,
     "identity": "sonde-paris-01.example.internal",
     "caRef": "ac-radius-example"
   }
   ```

3. Générez la clé et la demande de certificat sur la sonde (`POST /api/v1/system/dot1x/csr`). La clé privée est générée sur la sonde et ne la quitte jamais. Le sujet de la demande reprend l'identité 802.1X (`dot1x.identity`). Le type de clé se choisit dans le corps de la requête : `{"keyType": "ec-p256"}` (par défaut) ou `{"keyType": "rsa-2048"}` pour un serveur RADIUS plus ancien. La réponse contient la demande au format PEM.
4. Faites signer la demande par votre IGC, sous la forme d'un certificat d'authentification client reconnu par votre serveur RADIUS.
5. Importez le certificat signé sur la sonde par `PUT /api/v1/system/dot1x/credential`, au format PEM (certificat et chaîne) : la sonde l'associe à la clé générée lors de la demande. Si votre organisation fournit plutôt un fichier PKCS#12, importez-le par `PUT /api/v1/system/dot1x/credential`.
6. Une fois le certificat installé seulement, activez l'authentification : passez `enabled` à `true` dans la section `dot1x` (interface d'administration, ou `PUT /api/v1/settings/dot1x` avec l'en-tête `If-Match`), en conservant `identity` et `caRef`. Le corps correspond alors à l'exemple fictif ci-dessus.
7. Faites activer 802.1X sur le port de la sonde par l'équipe réseau, ou déplacez la sonde vers le port contrôlé prévu.

Résultat attendu : le commutateur autorise le port après l'authentification de la sonde ; la sonde reste joignable à son adresse.

Vérification : l'indicateur de santé `network` est à l'état `OK` ; le commutateur indique le port comme authentifié ; le journal du serveur RADIUS enregistre l'acceptation ; la métrique `sva.dot1x.certificate.expiry` est publiée (chapitre 17).

Erreurs possibles (deux voies) : en cas d'échec de l'authentification, la sonde effectue de nouvelles tentatives, affiche l'état de l'authentification sur la console HDMI et le signale par l'indicateur de santé `network`. Les causes habituelles sont une identité inconnue du serveur RADIUS, un certificat client non reconnu, expiré ou dépourvu de l'usage

d'authentification client, une AC du serveur RADIUS mal référencée (`caRef`) ou une heure incorrecte. Le diagnostic détaillé figure au chapitre 19 (§ 19.5).

Surveillance de l'expiration. Un certificat client expiré coupe l'accès de la sonde au réseau. La sonde publie la date d'expiration de son certificat client par la métrique `sva.dot1x.certificate.expiry` et émet un avertissement **30 jours** avant l'échéance. Renouvelez le certificat avant cette date, en reprenant les étapes 3 à 5 de la voie CSR. Pendant un renouvellement, l'identifiant en cours reste utilisé jusqu'à l'import du nouveau certificat : la nouvelle clé est conservée à part et ne remplace l'ancienne qu'à ce moment, sans interruption de l'accès au réseau. Intégrez cette échéance à vos alertes (chapitre 17).

Exception par adresse MAC (MAB). Lorsque 802.1X par certificat n'est pas envisageable, l'équipe réseau peut autoriser le port à partir de l'adresse MAC de la sonde. Pour l'offre boîtier complet, cette adresse figure sur l'étiquette du boîtier (§ 5.5.3). Pour l'offre microSD préconfigurée, elle dépend du Raspberry Pi 5 que vous fournissez : relevez-la sur le commutateur (adresse apprise sur le port) ou dans la table du serveur DHCP, lors d'un premier raccordement à un port non contrôlé. Aucun réglage n'est nécessaire sur la sonde.

> **Attention** — Une adresse MAC peut être usurpée par tout équipement raccordé au même port. Réservez MAB à la mise en service ou aux réseaux où votre politique de sécurité l'admet, et limitez les droits réseau associés au port (VLAN de supervision, § 7.10.2).

7.3 Résolution DNS

Le DNS (*Domain Name System*) traduit les noms de domaine en adresses IP. La sonde l'utilise :

- pour joindre les ressources supervisées par leur nom ;
- pour joindre les destinations d'export, les serveurs NTP et le dépôt de mises à jour ;
- dans les contrôles de type `dns`, qui vérifient eux-mêmes le bon fonctionnement de la résolution (chapitre 10).

Recommandations :

1. Configurez **deux serveurs DNS** au moins, pour que la défaillance de l'un n'entraîne pas une cascade de faux incidents.
2. Utilisez les **mêmes serveurs DNS que les utilisateurs** dont vous souhaitez reproduire l'expérience. Une sonde qui résout les noms autrement que les postes utilisateurs peut indiquer `OK` alors que les utilisateurs rencontrent un problème, ou l'inverse.
3. Si votre organisation utilise un DNS différent selon les zones (« split DNS »), vérifiez que les noms internes et externes à superviser sont résolus comme attendu depuis le réseau de la sonde.

> **Information** — Un contrôle de type `dns` peut interroger des serveurs DNS précis, différents de ceux configurés pour la sonde. C'est utile pour superviser individuellement chaque serveur DNS de l'organisation (chapitre 10).

7.4 Synchronisation de l'heure

L'heure exacte est indispensable au bon fonctionnement de la sonde :

Usage	Conséquence d'une heure incorrecte
Horodatage des résultats	Historique incohérent ; corrélation impossible avec les journaux des applications

Usage	Conséquence d'une heure incorrecte
Planification des contrôles par expression cron et fenêtres d'activité	Contrôles exécutés aux mauvais moments
Fenêtres de maintenance des mises à jour (chapitre 20, § 20.4.8)	Activation d'une mise à jour en dehors de la plage prévue
Vérification des certificats TLS	Rejet de certificats valides, ou acceptation de certificats expirés
Vérification de la fraîcheur des mises à jour en ligne (date d'expiration des métadonnées, chapitre 20, § 20.4.5)	Refus de métadonnées valides, ou acceptation de métadonnées expirées
Export vers les plateformes d'observabilité	Données rejetées ou mal placées dans le temps
Journal d'audit	Traçabilité compromise

Le Raspberry Pi 5 intègre une horloge temps réel. Sans pile optionnelle, elle ne conserve pas l'heure lorsque la carte est hors tension. La sonde s'appuie donc sur le protocole **NTP** (*Network Time Protocol*) pour obtenir l'heure à chaque démarrage, puis la maintenir. Sur un port soumis à 802.1X, l'accès NTP n'existe qu'après l'authentification EAP-TLS, qui exige elle-même une heure correcte : les chemins de pré-réglage de l'horloge sont décrits au § 7.2.6.

Recommandations :

1. Configurez au moins deux serveurs NTP internes de votre organisation. À défaut de configuration, la sonde utilise les serveurs NTP fournis par DHCP (option 42), sinon une zone fournisseur du *NTP Pool*, indiquée dans les notes de version.
2. Autorisez le flux UDP 123 de la sonde vers ces serveurs (§ 7.6).
3. Vérifiez l'état de synchronisation sur la page d'état de santé après toute modification.

La sonde enregistre et exporte toutes les dates en temps universel coordonné (UTC). Le fuseau horaire configuré sert uniquement à l'affichage et à l'interprétation des **fenêtres de maintenance des mises à jour**, exprimées en heure locale de la sonde (chapitre 20, § 20.4.8). Les expressions cron et les plages d'activité des contrôles ne dépendent pas de ce réglage : elles sont interprétées dans le fuseau indiqué par leur propre planification (propriété `timezone`, UTC par défaut, chapitre 12, § 12.2.3).

> **Information** — Une modification du fuseau horaire recalcule les activations de mises à jour déjà planifiées dans une fenêtre de maintenance, dont les bornes sont des heures locales, et les annonce de nouveau. Si, après ce recalcul, le préavis restant devient insuffisant, l'activation est reportée à la fenêtre de maintenance suivante. Une activation manuelle programmée par `scheduledAt` désigne un instant absolu, exprimé en UTC (RFC 3339) : elle reste fixée à cet instant, qui n'est ni recalculé ni reporté. Configurez vos outils de mise en sourdine des alertes dans le même fuseau que la sonde (chapitre 20, § 20.4.10).

> **Attention** — Une sonde dont l'heure n'est pas synchronisée le signale dans ses indicateurs de santé : l'indicateur `clock` reste à l'état `WARNING` jusqu'à la synchronisation. Au démarrage, les contrôles attendent la synchronisation de l'heure pendant 5 minutes au plus, puis s'exécutent même si elle n'est pas obtenue ; leurs vérifications TLS peuvent alors échouer.

7.5 Proxy sortant

De nombreuses organisations imposent un **proxy** pour les flux sortants vers Internet. Un proxy est un serveur intermédiaire qui relaie les requêtes web pour le compte des postes et des serveurs.

La sonde prend en charge un proxy HTTP(S) utilisant la méthode `CONNECT`, avec une authentification Basic facultative, et une **liste d'exclusions** pour les destinations joignables directement.

7.5.1 Quels flux passent par le proxy ?

Flux	Passage par le proxy
Exports vers des plateformes SaaS (Datadog, Dynatrace, Grafana Cloud)	Oui, sauf exclusion
Dépôt de mises à jour de l'éditeur	Oui, sauf exclusion
Contrôles HTTP et API, scénarios navigateur	Oui, sauf exclusion ; c'est souvent le comportement souhaité pour reproduire l'accès des utilisateurs à Internet. Chaque contrôle `http`, `api` ou `browser` peut contourner le proxy global (`"proxy": "none"`), mais pas en définir un autre
Contrôles ICMP, TCP, DNS, bases de données	Non : ces protocoles ne transitent pas par un proxy HTTP
NTP, DNS de la sonde	Non

7.5.2 Configurer la liste d'exclusions

La liste d'exclusions doit contenir toutes les destinations internes joignables directement : domaines internes (sous forme de suffixe, par exemple `.example.internal`), plages d'adresses internes, collecteur OpenTelemetry interne.

Exemple (fichier de provisionnement) :

```yaml
proxy:
  url: http://proxy.example.internal:3128
  noProxy: [".example.internal", "192.0.2.0/24", "198.51.100.0/24"]
```

✓ **Conseil** — Pour superviser une application interne **telle que la voient les utilisateurs**, reproduisez leur chemin réseau. Si les postes utilisateurs passent par le proxy pour joindre cette application, la sonde doit y passer aussi.

7.5.3 Inspection TLS par le proxy

Certains proxys déchiffrent les flux HTTPS pour les inspecter (« inspection TLS »). Ils présentent alors à la sonde des certificats émis par une autorité de certification interne. Pour que la sonde accepte ces certificats, importez cette autorité dans le magasin de confiance de la sonde (§ 7.7.3). Sinon, les contrôles et les exports échouent avec une erreur `tls_error`.

! **Attention** — Si le proxy authentifie ses utilisateurs, créez un compte technique dédié à la sonde (§ 7.8.3). Le mot de passe de ce compte est un secret : ne le placez pas dans le fichier de provisionnement (§ 6.3.3). Enregistrez-le comme secret de la sonde (chapitre 9) ; la configuration du proxy le référence par son nom (propriété `passwordSecret`), sans jamais contenir sa valeur.

7.6 Flux réseau et pare-feu

7.6.1 Matrice des flux

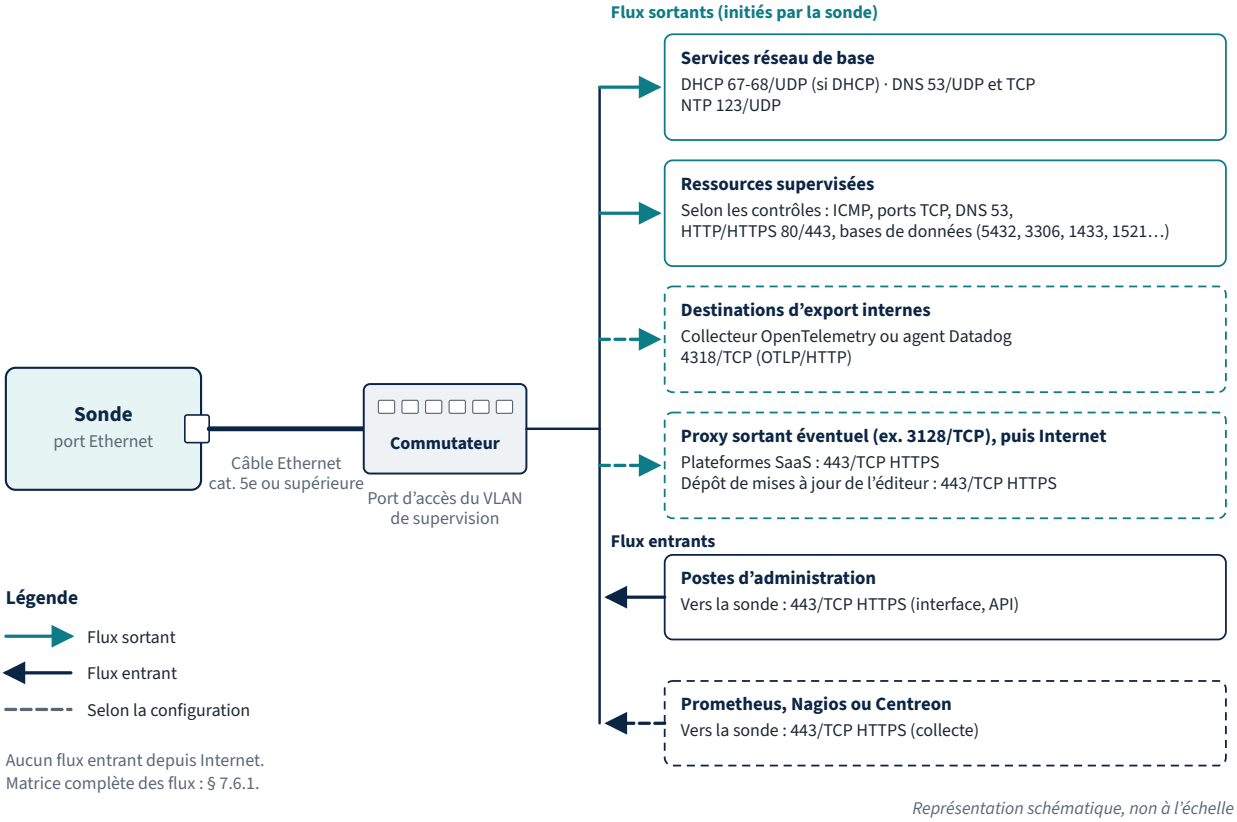


Figure 7-1 — Raccordement réseau de la sonde et principaux flux (le détail figure dans la matrice des flux ci-dessous)

Le tableau suivant recense l'ensemble des flux de la sonde. Transmettez-le à l'équipe chargée des pare-feu. La colonne « Obligatoire » distingue les flux indispensables des flux liés à une fonction facultative.

Sens	Source	Destination	Port / protocole	Usage	Obligatoire
Entrant	Postes d'administration	Sonde	443/TCP (HTTPS)	Interface d'administration, API	Oui
Entrant	Serveur Prometheus	Sonde	443/TCP (HTTPS)	Collecte de `/metrics`	Si Prometheus en mode collecte
Entrant	Serveur Nagios ou Centreon	Sonde	443/TCP (HTTPS)	Plugin `check_sva`	Si Nagios ou Centreon
Entrant	Postes d'administration	Sonde	80/TCP (HTTP)	Redirection permanente vers HTTPS, aucun contenu servi	Non (désactivable)
Entrant	Postes du réseau local	Sonde	5353/UDP (mDNS)	Résolution du nom `.local` de la sonde ; annonce active par défaut	Non (désactivable dans la section `network`)
Entrant	Postes d'administration autorisés	Sonde	22/TCP (SSH)	Maintenance avancée	Non : désactivé par défaut ; activation temporaire, 4 heures par défaut, 72 heures au maximum (§ 7.9.4)

Sens	Source	Destination	Port / protocole	Usage	Obligatoire
Couche 2 (sans flux IP)	Sonde	Commutateur (authentificateur 802.1X)	EAPOL (*EAP over LAN*) : trames Ethernet de type 0x888E, aucun port ni adresse IP	Authentification 802.1X EAP-TLS du port ; le commutateur relaie vers le serveur RADIUS	Si le port est contrôlé par 802.1X (§ 7.2.6)
Entrant et sortant	Sonde	Routeurs et voisins du lien local	ICMPv6 (découverte des voisins, NDP, dont les annonces de routeur utilisées en mode `slaac`)	Adressage et résolution d'adresses IPv6	Si l'IPv6 est activé (§ 7.2.5)
Sortant	Sonde	Serveurs DHCP	67-68/UDP	Adressage	Si DHCP
Sortant	Sonde	Serveurs DNS	53/UDP et 53/TCP	Résolution de noms	Oui
Sortant	Sonde	Serveurs NTP	123/UDP	Synchronisation de l'heure	Oui
Sortant	Sonde	Dépôt de mises à jour de l'éditeur, ou miroir interne	443/TCP (HTTPS)	Mises à jour en ligne : vérification et téléchargement des images signées ; adresse par défaut `https://updates-synovex.incitypark.fr`, inscrite dans le profil éditeur	Si mises à jour en ligne
Sortant	Sonde	Proxy de l'entreprise	Port du proxy (par exemple 3128/TCP)	Relais des flux HTTP(S) sortants (§ 7.5)	Si proxy
Sortant	Sonde	Plateformes SaaS (Datadog, Dynatrace, Grafana Cloud)	443/TCP (HTTPS, OTLP/HTTP)	Export OpenTelemetry	Si configurées
Sortant	Sonde	Collecteur OpenTelemetry ou Agent Datadog du client	4318/TCP (OTLP/HTTP) ou 4317/TCP (OTLP/gRPC), selon le protocole de la destination (chapitre 18, § 18.1.4)	Export OpenTelemetry	Si configurés
Sortant	Sonde	Proxy inverse placé devant le récepteur OTLP de Prometheus	Port du proxy inverse (par exemple 443/TCP, HTTPS)	Export des métriques en mode envoi (chapitre 18, § 18.5.3)	Si configuré
Sortant	Sonde	Ressources supervisées	Selon les contrôles : ICMP ou ICMPv6 ; ports TCP des services ; 53 ; 80 et 443 ; SGBD : 5432 (PostgreSQL), 3306 (MySQL/MariaDB), 1433 (SQL Server), 1521 (Oracle)	Contrôles	Oui, selon les contrôles
Sortant	Sonde	Console centrale	443/TCP (HTTPS, authentification mutuelle mTLS)	Gestion multi-sondes	Non : évolution prévue, facultative

Lorsque l'IPv6 est activé, les flux de ce tableau s'entendent en IPv4 et en IPv6, selon la famille d'adresses utilisée.

> **Information** — L'édition est vérifiée hors ligne, à partir du certificat d'édition inscrit sur la carte : aucun flux vers un serveur de licences n'est nécessaire. La sonde n'envoie **aucune donnée à l'éditeur** (pas de télémétrie) : ses seuls échanges avec lui sont la vérification et le téléchargement des mises à jour, si vous les activez.

> **Évolution prévue** — La console centrale facultative (chapitre 9, § 9.14) utilisera une connexion sortante initiée par la sonde, sans ouverture entrante. Aucun flux vers elle n'est nécessaire aujourd'hui.

7.6.2 Règles recommandées

1. **Entrant** : limitez l'accès au port 443 de la sonde aux seuls postes d'administration, serveur Prometheus et serveur Nagios/Centreon. Refusez tout le reste.
2. **Sortant vers les ressources supervisées** : ouvrez uniquement les destinations et ports utilisés par vos contrôles. Une ouverture large (« tout le réseau interne ») facilite la configuration mais accroît l'impact d'une éventuelle compromission de la sonde.
3. **Sortant vers Internet** : limitez-le aux destinations nécessaires (plateformes SaaS, dépôt de mises à jour), de préférence via le proxy.
4. **ICMP** : si vous utilisez des contrôles de type `icmp`, autorisez les requêtes et réponses ICMP *Echo* entre la sonde et les cibles concernées, ainsi que leurs équivalents ICMPv6 pour les cibles jointes en IPv6. Si l'IPv6 est activé, ne filtrez pas l'ICMPv6 de découverte des voisins sur le réseau local de la sonde.
5. Journalisez les flux refusés depuis et vers la sonde : ces journaux sont précieux pour diagnostiquer un contrôle en échec (chapitre 19).

7.6.3 Pare-feu local de la sonde

En complément de vos pare-feu réseau, la sonde dispose d'un pare-feu local (nftables), **actif par défaut**. Il n'autorise en entrée que les ports nécessaires : 443/TCP, 80/TCP (redirection vers HTTPS, § 7.7.1), 5353/UDP (mDNS), et 22/TCP lorsque SSH est activé. Une liste facultative de réseaux autorisés, en notation CIDR (par exemple `192.0.2.0/24`), peut être configurée dans les paramètres de sécurité (section `security`) ; elle s'applique aux ports 443 et 22. Lorsque l'IPv6 est activé, le pare-feu local applique les mêmes règles en IPv6 (§ 7.2.5).

> → **Évolution prévue** — Pour l'édition Enterprise, la spécification prévoit un filtrage des connexions sortantes de l'environnement d'exécution des scénarios, limitant celles-ci aux destinations déclarées par l'administrateur.

7.7 Certificats et protocoles sécurisés

7.7.1 Protocoles utilisés

Échange	Protocole	Protection
Administration, API, `/metrics`	HTTPS	TLS 1.2 minimum, TLS 1.3 privilégié
Export vers les plateformes	HTTPS (OTLP/HTTP) ou OTLP/gRPC	TLS ; HTTP non chiffré accepté uniquement vers un collecteur interne si vous le configurez ainsi
Accès de maintenance	SSH	Chiffré ; authentification par clé publique uniquement, mot de passe interdit
Mises à jour	HTTPS	TLS, et signature de l'image de mise à jour par l'éditeur, vérifiée avant installation (chapitre 20)
Contrôles	Ceux des ressources supervisées	La sonde vérifie les certificats des cibles HTTPS par défaut

Le port HTTP 80 sert uniquement à une redirection permanente vers HTTPS : aucun contenu n'y est servi. Cette redirection peut être désactivée.

7.7.2 Remplacer le certificat de la sonde

Le certificat autosigné généré au premier démarrage protège les échanges, mais n'est reconnu par aucun navigateur ni outil. Il impose une vérification manuelle de l'empreinte (§ 6.4.4) et la diffusion du certificat aux outils qui interrogent la sonde. Il est donc recommandé de le remplacer par un certificat émis par l'**autorité de certification (AC) interne** de votre organisation.

Objectif : faire reconnaître la sonde par les navigateurs et outils de votre parc.

Prérequis :

- nom DNS de la sonde enregistré (exemple : `sonde-paris-01.example.internal`) ;
- accès à l'AC interne de votre organisation ;
- compte ou jeton de rôle `admin`.

Étapes :

1. Obtenez un certificat serveur pour le nom DNS de la sonde, en incluant ce nom dans l'extension *Subject Alternative Name*. Deux modes sont proposés : la **génération de la clé et de la demande de certificat (CSR) sur la sonde**, recommandée car la clé privée ne quitte jamais l'appareil, ou l'import d'une clé privée générée ailleurs.
2. Importez le certificat, la chaîne de certification et, le cas échéant, la clé privée, depuis l'interface ou par l'API. Si la demande a été générée sur la sonde (`POST /api/v1/system/certificate/csr`), seul le certificat signé et sa chaîne sont à importer. L'import se fait par `PUT /api/v1/system/certificate`. Formats acceptés : PEM (certificat, chaîne et clé), ou PKCS#12 protégé par un mot de passe.
3. Reconnectez-vous à l'interface en utilisant le nom DNS.

Résultat attendu : le navigateur affiche une connexion sécurisée sans avertissement.

Vérification : `curl https://sonde-paris-01.example.internal/health` réussit sans option `--cacert` sur un poste qui fait confiance à votre AC interne.

> **Attention** — Anticipez l'expiration du certificat : une sonde dont le certificat a expiré reste fonctionnelle, mais les outils qui la collectent (Prometheus, Nagios) échoueront. Planifiez son renouvellement dans vos processus habituels.

7.7.3 Faire confiance à une AC interne pour les contrôles et les exports

Indépendamment de son propre certificat, la sonde doit pouvoir **vérifier** les certificats des services qu'elle contacte : applications supervisées, collecteur OpenTelemetry interne, proxy avec inspection TLS. Si ces certificats sont émis par une AC interne, importez le certificat de cette AC, au format PEM, dans le magasin de confiance de la sonde, géré dans la section `trust` des paramètres (chapitre 9, § 9.9). Ce magasin s'ajoute au magasin système et s'applique aux contrôles `http`, `api`, `tcp` (avec TLS) et `browser`, ainsi qu'aux exports.

Les AC importées s'appliquent par défaut à tous les contrôles concernés. Un contrôle HTTP peut en outre restreindre sa confiance à une seule AC importée (propriété `tls.caRef`, chapitre 10).

> **Avertissement** — La désactivation de la vérification des certificats (`tls.verify: false` dans un contrôle, `insecureSkipVerify` dans une destination d'export) doit rester exceptionnelle, temporaire et documentée. Elle supprime la protection contre l'interception des échanges, y compris des identifiants des comptes de test.

7.8 Identifiants, secrets et comptes techniques

7.8.1 Ce que la sonde doit connaître

Pour superviser vos applications en profondeur, la sonde a besoin d'identifiants : comptes de test pour se connecter aux applications web, comptes de lecture pour interroger les bases de données, jetons pour appeler les API, clés d'API des plateformes d'observabilité. Ces informations sont des **secrets**.

La sonde protège les secrets de la manière suivante :

Mesure	Effet
Stockage chiffré	Les secrets sont chiffrés sur la microSD
Écriture seule	Un secret enregistré ne peut jamais être relu, ni par l'interface ni par l'API ; il peut seulement être remplacé ou supprimé
Déclaration par contrôle	Chaque contrôle déclare les secrets qu'il utilise ; seuls ceux-ci lui sont transmis lors de l'exécution
Référencement	Les configurations ne contiennent jamais la valeur d'un secret, mais une référence de la forme `${secret:NOM}`
Masquage	Les valeurs des secrets sont masquées dans les journaux et les messages d'erreur
Exclusion des exports	Les exports de configuration excluent les secrets, ou les chiffrent par une phrase de passe

La procédure de création, de rotation et de suppression des secrets est décrite au chapitre 9.

> **Attention** — Sans composant matériel de sécurité, la clé qui protège les secrets réside sur la même microSD que les secrets eux-mêmes. Le chiffrement protège contre la copie de la base de données, mais **pas contre le vol de la microSD entière**. La sécurité physique de la sonde (§ 7.11) et le moindre privilège des comptes (§ 7.8.2) restent donc essentiels.

> **Attention** — Les captures d'écran produites en cas d'échec d'un scénario (chapitre 17) peuvent afficher des données présentes à l'écran. Les champs de mot de passe sont masqués par le navigateur, mais une page peut réafficher une donnée sensible. Utilisez des comptes de test sans données réelles.

7.8.2 Principe du moindre privilège

Le **principe du moindre privilège** consiste à n'accorder à un compte que les droits strictement nécessaires à sa fonction. Appliqué à la sonde, il limite les conséquences d'une fuite d'identifiants ou d'une erreur de configuration.

Usage	Recommandation
Comptes de test applicatifs (scénarios navigateur, API)	Comptes **dédiés à la supervision**, nominativement identifiables (par exemple `test-supervision@example.com`), sans droits d'administration, sans accès à des données réelles, sans moyen de paiement réel
Comptes de bases de données	Compte **en lecture seule**, limité aux tables ou vues nécessaires ; idéalement, une vue dédiée qui expose uniquement les indicateurs à contrôler
Jetons d'API des applications	Portée minimale (lecture seule), durée de validité limitée
Clés d'API des plateformes d'observabilité	Clés limitées à l'**ingestion** des données (par exemple, pour Dynatrace, un jeton disposant uniquement des droits d'ingestion des signaux exportés)
Comptes d'accès à la sonde	Rôle minimal adapté à chaque usage (§ 7.9.1)

Exemple de compte en lecture seule pour un contrôle de base de données PostgreSQL (fictif) :

```sql
-- Compte de supervision en lecture seule, limité à une vue dédiée (exemple PostgreSQL)
CREATE ROLE supervision_ro LOGIN PASSWORD '********';
CREATE VIEW reporting.v_supervision_commandes AS
  SELECT count(*) AS nb_commandes_jour
  FROM ventes.commandes
  WHERE created_at >= current_date;
GRANT USAGE ON SCHEMA reporting TO supervision_ro;
GRANT SELECT ON reporting.v_supervision_commandes TO supervision_ro;
```

Information — Exposer une vue plutôt que les tables évite de donner à la sonde l'accès aux données détaillées : elle ne lit que l'indicateur agrégé dont elle a besoin. Les contrôles de bases de données sont décrits au chapitre 11.

7.8.3 Comptes techniques dédiés

Créez des comptes techniques distincts pour chaque usage et chaque sonde lorsque c'est possible :

- un compte par application supervisée, plutôt qu'un compte partagé entre toutes ;
- un compte de proxy dédié à la sonde ;
- une clé d'API d'ingestion par plateforme et par site.

Vous pourrez ainsi révoquer un accès sans interrompre les autres, et identifier précisément l'origine des connexions dans les journaux de vos applications.

Conseil — Informez les équipes applicatives de l'existence des comptes de test de la sonde. Elles éviteront de les désactiver lors d'une revue des comptes, et pourront exclure leur activité des statistiques métier.

7.8.4 Rotation des secrets

Renouvelez les secrets selon la politique de votre organisation, et immédiatement en cas de suspicion de compromission. La sonde conserve la date de dernière rotation de chaque secret (chapitre 9). Pour une rotation sans interruption de la supervision :

1. Créez le nouveau mot de passe dans l'application cible (ou un second jeton).
2. Remplacez la valeur du secret dans la sonde.
3. Déclenchez une exécution immédiate des contrôles concernés et vérifiez leur état.
4. Désactivez l'ancien mot de passe ou jeton dans l'application cible.

7.9 Sécurisation des accès administrateur

7.9.1 Comptes et rôles

L'accès à la sonde est contrôlé par des **rôles**. Chaque compte utilisateur et chaque jeton d'API reçoit un rôle.

Rôle	Droits	Usage type
`admin`	Tous les droits : utilisateurs, jetons, secrets, réseau, mises à jour, scripts et plugins personnalisés	Administrateurs de la sonde (2 personnes au moins, comptes nominatifs)

Rôle	Droits	Usage type
`operator`	Création et modification des contrôles déclaratifs et des plans, activation, désactivation, exécution immédiate, consultation. Ni code exécutable ; référence d'un secret existant possible, sans le lire, le créer ni le modifier	Équipe d'exploitation
`viewer`	Lecture seule des états, résultats et artefacts	Équipes applicatives, support
`metrics`	Lecture de `/metrics` et de l'état des contrôles uniquement	Serveur Prometheus, plugin Nagios/Centreon

Recommandations :

1. Créez des **comptes nominatifs**, un par personne. N'utilisez pas de compte partagé.
2. Limitez le nombre d'administrateurs, en en conservant au moins deux pour éviter une perte d'accès.
3. Le téléversement de scripts et de plugins personnalisés est réservé au rôle `admin`, car il revient à exécuter du code sur le réseau où la sonde est installée.
4. Revoyez périodiquement la liste des comptes et des jetons.

La gestion des comptes est décrite au chapitre 9. Les comptes sont **locaux** à la sonde : l'authentification par l'annuaire d'entreprise (LDAP, Active Directory) ou par un service d'authentification unique n'est pas disponible.

> → **Évolution prévue** — L'authentification par OpenID Connect est prévue dans une version ultérieure (chapitre 9, § 9.4.2).

7.9.2 Jetons d'API

Les outils (Prometheus, Nagios, scripts d'automatisation) s'authentifient par **jeton d'API**. Bonnes pratiques :

- un jeton par outil et par usage, avec le rôle minimal (`metrics` pour Prometheus et Nagios) ;
- une date d'expiration ;
- un stockage dans le coffre-fort ou le gestionnaire de secrets de l'outil, jamais dans un dépôt de code ;
- une révocation immédiate en cas de doute.

Le jeton n'est affiché qu'une fois, à sa création ; la sonde n'en conserve qu'une empreinte.

7.9.3 Protection de l'interface d'administration

Mesure	Description
HTTPS uniquement	Aucune donnée d'authentification ne transite en clair
Protection contre les tentatives répétées	Après 5 échecs de connexion consécutifs, blocage pendant 15 minutes, par compte et par adresse source
Sessions sécurisées	Cookies de session protégés ; expiration après 30 minutes d'inactivité, et en tout état de cause après 12 heures
Restriction réseau	Filtrage par pare-feu (§ 7.6.2) et liste d'adresses autorisées (§ 7.6.3)
Poste d'administration	Utiliser un poste maîtrisé, à jour, de préférence un poste d'administration dédié

7.9.4 Accès SSH

L'accès SSH (*Secure Shell*) donne un accès en ligne de commande au système de la sonde. Il n'est **pas nécessaire** à l'exploitation courante : toutes les opérations prévues sont disponibles dans l'interface d'administration et par l'API. Il est donc **désactivé par défaut**.

Il peut être activé temporairement pour un diagnostic avancé, par exemple à la demande du support. L'éditeur ne dispose d'**aucun accès permanent** à la sonde et ne peut pas ouvrir cet accès lui-même : seul un administrateur de la sonde (rôle `admin`) peut l'activer, et toujours pour une **durée limitée**.

Objectif : ouvrir un accès SSH temporaire et maîtrisé.

Prérequis : compte de rôle `admin` ; paire de clés SSH de l'intervenant ; règle de pare-feu autorisant le port 22/TCP depuis le seul poste de l'intervenant ; accord du RSSI selon votre politique.

Étapes :

1. Activez l'accès SSH dans les paramètres de la sonde, section `ssh` : rubrique « Accès de maintenance SSH » de l'interface d'administration, ou `PUT /api/v1/settings/ssh` (champs `enabled`, `enabledUntil` et `authorizedKeys`, avec l'en-tête `If-Match`, § 7.2.4). La date de fin `enabledUntil` est **obligatoire** à l'activation : la durée est de 4 heures par défaut et de 72 heures au maximum.
2. Déposez la clé publique de l'intervenant. L'authentification par mot de passe est interdite : seule l'authentification par clé publique est acceptée. Compte système utilisé : un compte dédié, non privilégié, dont toute élévation de droits est journalisée ; nom du compte : `sva-maint`.
3. Réalisez l'intervention.
4. **Désactivez SSH** dès la fin de l'intervention, sans attendre l'échéance, et retirez la clé publique.
5. Supprimez la règle de pare-feu temporaire.

Exemple de section `ssh` pour une activation temporaire (valeurs fictives) :

```json
{
  "enabled": true,
  "enabledUntil": "2025-11-20T18:00:00Z",
  "authorizedKeys": [
    "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIExempleFictifDeCleIntervenant intervenant@example.internal"
  ]
}
```

Résultat attendu : SSH est accessible avec la clé déposée, depuis le seul poste autorisé, jusqu'à la date `enabledUntil`. À cette échéance, la sonde **désactive automatiquement** SSH et **ferme les sessions en cours**.

Vérification : le journal d'audit contient l'activation de SSH, chaque connexion SSH et la désactivation, manuelle ou automatique.

Erreurs possibles : connexion refusée si la clé présentée ne figure pas dans `authorizedKeys`, si le pare-feu réseau ou la liste des réseaux autorisés (§ 7.6.3) bloque le port 22, si la date `enabledUntil` est dépassée ou si SSH a été verrouillé définitivement (ci-dessous). Le diagnostic par SSH est traité au chapitre 19 (§ 19.2.4).

> **Avertissement** — Une intervention en ligne de commande peut modifier le système en dehors des mécanismes de la sonde et compromettre sa stabilité, sa sécurité ou ses mises à jour. N'effectuez que les opérations documentées par l'éditeur ou demandées par le support.

Verrouillage définitif de SSH. Si votre politique de sécurité interdit tout accès en ligne de commande, vous pouvez verrouiller définitivement SSH. Une fois verrouillé, SSH ne peut plus être activé, ni par l'interface ni par l'API, jusqu'à une réinitialisation usine (chapitre 20, § 20.10).

Objectif : supprimer durablement toute possibilité d'activer SSH sur la sonde.

Prérequis : compte ou jeton de rôle `admin` ; décision validée par le RSSI ; nom d'hôte exact de la sonde.

Étapes :

1. Dès la mise en service, ajoutez au fichier de provisionnement la ligne `locked: true` dans la section `ssh` (§ 6.3.3) ; ou, sur une sonde en service, appelez `POST /api/v1/settings/ssh/lock` en confirmant l'opération par le nom d'hôte de la sonde :

   ```bash
   # Verrouillage définitif de SSH (rôle admin). Le champ confirm doit contenir
   # le nom d'hôte exact de la sonde ; sinon la demande est refusée.
   # Préparation du poste (SONDE, CA, fonction sapi) : § 9.3.3
   sapi -X POST -H 'Content-Type: application/json' \
        --data '{"confirm": "sonde-paris-01"}' \
        "$SONDE/api/v1/settings/ssh/lock"
   ```

2. Vérifiez le résultat (ci-dessous).

Résultat attendu : la propriété `locked` de la section `ssh` vaut `true` ; toute demande d'activation de SSH est refusée.

Vérification : `GET /api/v1/settings/ssh` ; le journal d'audit enregistre le verrouillage.

Erreurs possibles : demande refusée si le nom d'hôte transmis dans `confirm` ne correspond pas exactement à celui de la sonde, ou si le compte n'a pas le rôle `admin`.

> ! **Avertissement** — Le verrouillage de SSH est **irréversible** : seule une réinitialisation usine, qui efface la configuration, permet de le lever. Une fois SSH verrouillé, le support ne pourra plus intervenir sur la sonde par SSH ; les diagnostics passeront uniquement par l'interface, l'API et l'archive de diagnostic (chapitre 19). La récupération du compte administrateur par code de récupération reste possible (chapitre 20).

7.10 Segmentation et positionnement réseau

7.10.1 Où placer la sonde ?

La sonde mesure ce qu'elle « voit » depuis son point de raccordement. Son emplacement dépend de la question à laquelle elle doit répondre.

Objectif	Positionnement recommandé
Mesurer l'expérience des utilisateurs d'un site	Réseau des postes utilisateurs de ce site, mêmes DNS, même proxy
Superviser des services internes au centre de données	Réseau d'administration ou de supervision du centre de données
Comparer la disponibilité depuis plusieurs sites	Une sonde par site, chacune avec un attribut `site` distinct (chapitre 18)
Vérifier l'accès à une application exposée sur Internet	Réseau disposant de l'accès Internet standard de l'organisation

> ✓ **Conseil** — Le déploiement de deux sondes sur des sites différents fournit des indices pour localiser une panne : un échec vu depuis les deux sites oriente vers l'application ou vers un composant commun, un échec vu depuis un seul vers ce qui est propre à ce site. Ces indices ne suffisent pas à conclure. Un composant réseau commun peut affecter les deux sites, et une réponse qui dépend de la région, de l'identité, du répartiteur de charge ou du DNS peut n'affecter qu'un seul point d'observation. Comparez les chemins, les cibles, les résolutions DNS, les politiques appliquées et les résultats détaillés des deux sondes avant d'attribuer l'origine à l'application ou au réseau (chapitre 3, § 3.2.1).

7.10.2 VLAN de supervision

Lorsque votre réseau est segmenté, il est recommandé de placer la sonde dans un **VLAN** (réseau local virtuel) dédié à la supervision, ou dans le VLAN d'administration existant. Les échanges avec les ressources supervisées sont alors contrôlés par les règles de filtrage entre VLAN (§ 7.6.2).

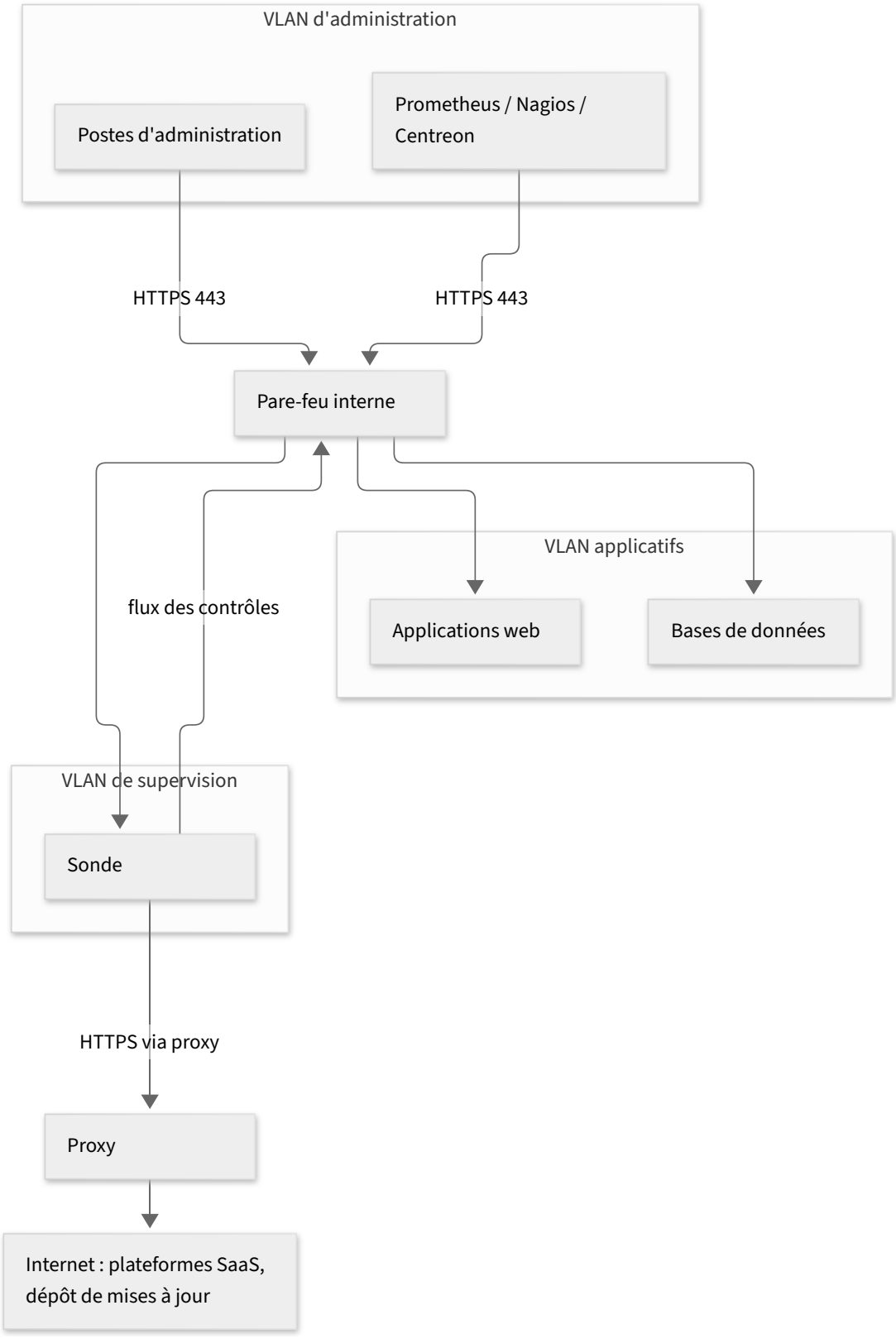


Schéma 7-2 — Sonde placée dans un VLAN de supervision

Ce découpage offre trois avantages :

- les flux d'administration et de collecte sont limités aux seuls outils autorisés ;
- les flux des contrôles sont explicitement autorisés, cible par cible ;
- une compromission éventuelle de la sonde ne donne pas un accès direct à l'ensemble du réseau.

> ! **Attention** — Le cloisonnement réseau modifie le chemin des flux. Vérifiez que la sonde, placée dans un VLAN dédié, mesure toujours l'expérience que vous souhaitez observer (§ 7.10.1). Si nécessaire, raccordez une seconde sonde dans le réseau des utilisateurs.

7.11 Sécurité physique

La microSD contient le système, la configuration et les secrets chiffrés de la sonde. Une personne ayant un accès physique à la sonde peut retirer la carte et en faire une copie.

Recommandations :

1. Installez la sonde dans un local ou une baie à accès contrôlé.
2. Ne laissez pas d'écran ni de clavier raccordé en permanence.
3. Tenez un inventaire des sondes (numéro de série, emplacement, adresse MAC).
4. En cas de vol ou de perte d'une sonde ou d'une microSD : changez sans délai **tous les secrets** que la sonde connaissait (comptes de test, comptes de bases de données, clés d'API d'ingestion), révoquez les jetons d'API et, s'il a été émis par votre AC interne, le certificat de la sonde ; révoquez le certificat client 802.1X de la sonde auprès de votre AC et retirez son exception MAB éventuelle ; puis retirez ses règles de pare-feu.
5. Avant de mettre au rebut ou de retourner une microSD, effacez-la de manière sécurisée ou détruisez-la (chapitre 20).

7.12 Journalisation et audit

7.12.1 Journal d'audit

La sonde tient un **journal d'audit** des opérations d'administration. Chaque événement comporte l'horodatage, l'auteur (compte ou jeton), l'adresse source, l'action, la ressource concernée et le résultat. Les modifications de configuration y figurent sans la valeur des secrets.

Événements journalisés :

- connexions réussies et échouées ;
- création, modification et suppression de contrôles, plans d'exécution, destinations d'export, comptes et jetons ;
- création, rotation et suppression de secrets (sans leur valeur) ;
- téléversement de scripts et de plugins ;
- modifications des paramètres réseau et de sécurité, activation, désactivation et verrouillage de SSH, connexions SSH ;
- gestion de l'identifiant client 802.1X ;
- mises à jour, réinitialisations, imports et exports de configuration.

Le journal est consultable par les administrateurs dans l'interface et par l'API (`GET /api/v1/audit-events`). Durée de conservation par défaut : 365 jours.

7.12.2 Centralisation dans le SIEM

Pour une conservation conforme à votre politique et une corrélation avec les autres événements de sécurité, transmettez les événements d'audit à votre SIEM (*Security Information and Event Management*). La sonde peut exporter ses journaux, y compris les événements d'audit, au format OpenTelemetry (chapitre 18). L'export des événements d'audit, sous forme de logs OTLP, est facultatif et désactivé par défaut ; il s'active destination par destination (`logs.audit: true`). Pour une extraction ponctuelle, `GET /api/v1/audit-events` accepte les filtres `from`, `to`, `actor` et `action`.

7.12.3 Journaux système

Les journaux système de la sonde sont conservés **en mémoire** par défaut, afin de limiter l'usure de la microSD : ils sont perdus au redémarrage. Pour un diagnostic portant sur une période incluant un redémarrage, la conservation sur disque peut être activée temporairement, par le paramètre `logging.persistentUntil`, qui fixe sa date de fin ; elle est désactivée automatiquement au bout de 72 heures au plus. Le diagnostic est traité au chapitre 19.

7.13 Recommandations aux RSSI

Le tableau suivant synthétise les points à valider avec la fonction sécurité avant la mise en production.

Domaine	Recommandation	Référence
Exposition	Aucun accès depuis Internet ; port 443 de la sonde ouvert aux seuls outils et postes autorisés	§ 7.1, § 7.6
Segmentation	Sonde dans un VLAN de supervision ou d'administration ; flux sortants limités aux cibles	§ 7.10
Contrôle d'accès au réseau	802.1X par certificat (EAP-TLS), clé générée sur la sonde ; MAB limité aux cas admis ; suivi de l'expiration du certificat client	§ 7.2.6
Certificats	Certificat de la sonde émis par l'AC interne ; vérification des certificats des cibles conservée	§ 7.7
Comptes	Comptes nominatifs ; rôles minimaux ; deux administrateurs au moins ; revue périodique	§ 7.9.1
Jetons	Un jeton par outil, rôle `metrics` pour la collecte, expiration, stockage en coffre-fort	§ 7.9.2
SSH	Désactivé ; activation temporaire (72 heures au plus), tracée et justifiée ; verrouillage définitif si la politique de sécurité l'exige	§ 7.9.4
Comptes de test	Dédiés, sans privilèges, sans données réelles ni moyens de paiement réels	§ 7.8.2
Bases de données	Comptes en lecture seule, vues dédiées	§ 7.8.2
Secrets	Rotation planifiée ; procédure de révocation en cas de perte de la sonde	§ 7.8.4, § 7.11
Scripts	Téléversement réservé aux administrateurs ; relecture des scripts avant mise en service	§ 7.9.1, chapitre 16
Traçabilité	Journal d'audit transmis au SIEM	§ 7.12
Mises à jour	Mode automatique ou processus manuel régulier ; suivi des correctifs de sécurité	Chapitre 20
Physique	Local à accès contrôlé ; inventaire ; procédure de mise au rebut	§ 7.11
Données personnelles	Captures d'écran limitées aux échecs, rétention courte, comptes de test sans données réelles	§ 7.8.1

Information — Les caractéristiques de sécurité détaillées du produit (isolation de l'exécution, chiffrement, signature des mises à jour) sont décrites au chapitre 3. Pour toute question relative à la sécurité du produit, contactez le support de l'éditeur (chapitre 21).

PARTIE III

Utilisation de la sonde

Chapitre 8 — Principes de fonctionnement

Chapitre 9 — Administration de la sonde

Chapitre 10 — Contrôles réseau et techniques

Chapitre 11 — Contrôles applicatifs et métier

Chapitre 12 — Planification et plans d'exécution

Chapitre 8 — Principes de fonctionnement

Ce chapitre présente les notions communes à tous les contrôles exécutés par la sonde : comment un contrôle est défini, quand il s'exécute, comment son résultat est calculé et comment l'interpréter. Ces principes s'appliquent aussi bien à une simple vérification ICMP qu'à un parcours navigateur en plusieurs étapes. Les chapitres suivants s'appuient sur eux :

- le chapitre 10 détaille les contrôles réseau et techniques ;
- le chapitre 11, les contrôles applicatifs et métier ;
- le chapitre 12, la planification ;
- les chapitres 13 à 16, les scénarios Puppeteer.

Les exemples de ce chapitre utilisent des noms de domaine et des adresses fictifs, réservés à la documentation (`example.com`, `192.0.2.0/24`).

8.1 La notion de contrôle

8.1.1 Définition

Un **contrôle** est l'unité de base de la supervision. Il décrit une vérification que la sonde doit réaliser de façon répétée :

- **quoi** vérifier : le type de contrôle et sa cible ;
- **comment** juger le résultat : les assertions, les règles et les seuils ;
- **quand** l'exécuter : la planification ;
- **dans quelles limites** : le délai maximal (timeout) et les nouvelles tentatives.

Chaque lancement d'un contrôle produit une **exécution**. Une exécution a un début, une durée, un **état** (`OK`, `WARNING`, `CRITICAL`, `UNKNOWN`) et, le cas échéant, une description de l'erreur rencontrée.

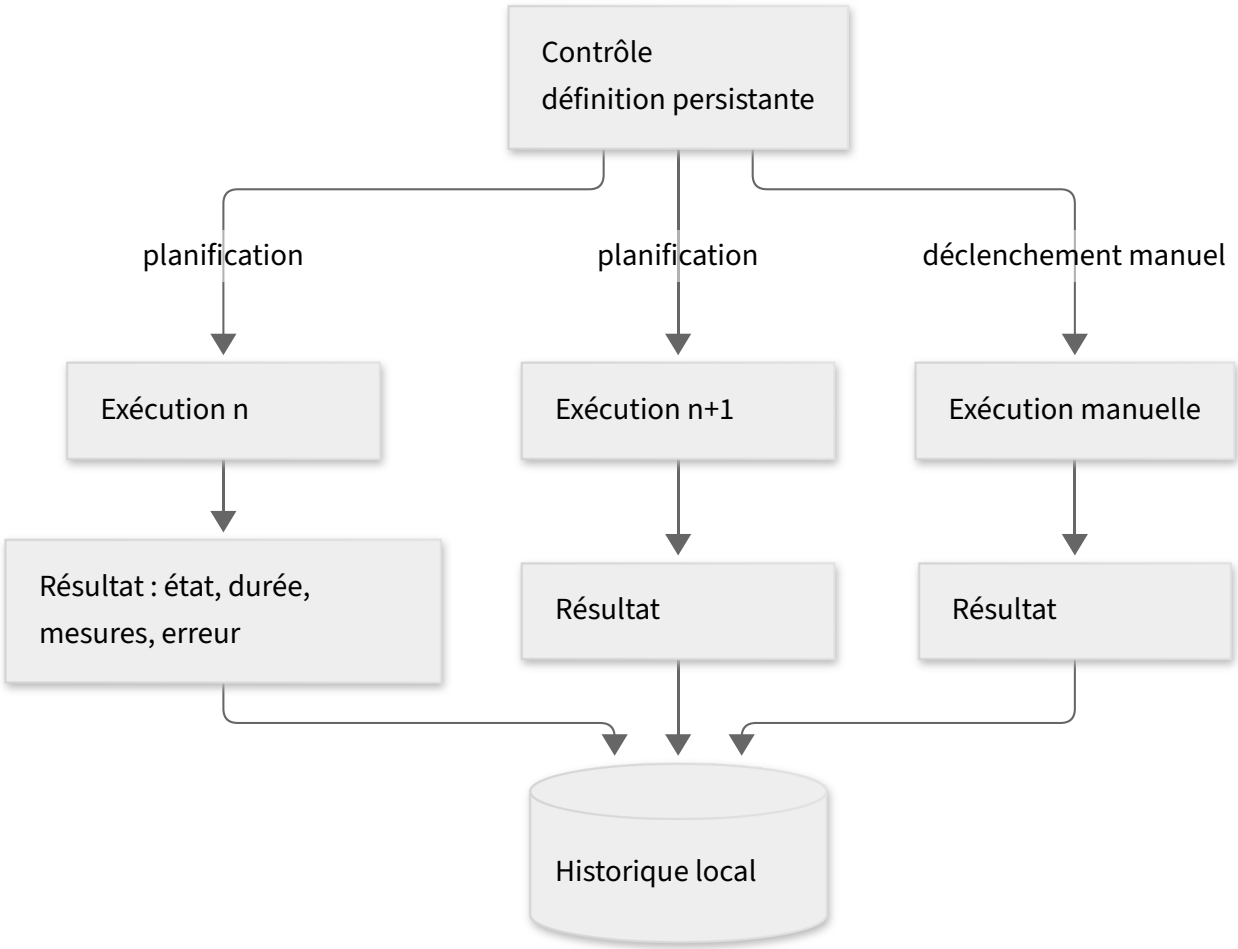


Schéma 8-1 — Un contrôle et ses exécutions

Les contrôles sont stockés **sur la sonde elle-même**. Ils continuent d'être exécutés quand aucune plateforme externe n'est joignable (§ 8.14).

8.1.2 Types de contrôles

La sonde prend en charge huit types de contrôles. Leur disponibilité dépend de l'édition acquise.

Type	Vérifie	Édition minimale	Chapitre
`icmp`	La joignabilité d'un équipement (ping), la latence et la perte de paquets	Essential	10
`tcp`	L'ouverture d'un port TCP et, en option, un échange simple	Essential	10
`dns`	La résolution d'un nom et la conformité de la réponse	Essential	10
`http`	Une requête HTTP ou HTTPS : code de retour, en-têtes, contenu, durée, certificat	Essential	10
`nagios`	L'exécution d'un plugin compatible Nagios	Essential (plugins fournis), Professional (plugins personnalisés)	10

Type	Vérifie	Édition minimale	Chapitre
`api`	Une suite de requêtes HTTP enchaînées, avec réutilisation de valeurs d'une réponse à l'autre	Professional	11
`database`	Le résultat d'une requête ou d'une procédure SQL au regard de règles	Professional	11
`browser`	Un parcours utilisateur dans un navigateur Chromium sans interface (headless)	Enterprise	13 à 16

8.1.3 Structure commune

Quel que soit son type, un contrôle partage un ensemble de propriétés communes. Il les complète par une section propre à son type (`http`, `database`, `browser`…).

Propriété	Rôle	Exemple
`id`	Identifiant stable, choisi par vous. Il apparaît dans les résultats, l'API et la télémétrie	`portail-accueil`
`name`	Libellé lisible	`Portail client : page d'accueil`
`type`	Type de contrôle (§ 8.1.2)	`http`
`enabled`	Contrôle actif ou suspendu	`true`
`tags`	Étiquettes libres (8 au maximum) servant au filtrage et au regroupement	`{ "application": "portail" }`
`schedule`	Planification (§ 8.3 et chapitre 12)	`{ "interval": "1m" }`
`timeout`	Durée maximale d'une exécution (§ 8.4)	`15s`
`retry`	Nouvelles tentatives avant de confirmer un échec (§ 8.5)	`{ "attempts": 1, "interval": "20s" }`
`thresholds`	Seuils de performance (§ 8.6)	`{ "warningDuration": "3s" }`
`secrets`	Liste des secrets que le contrôle est autorisé à utiliser (chapitre 9)	`["API_TOKEN"]`
`ipFamily`	Famille d'adresses utilisée pour joindre la cible : `auto` (valeur par défaut), `ipv4` ou `ipv6` (voir ci-dessous)	`auto`

Le paramètre `ipFamily` détermine la version du protocole IP employée pour joindre la cible :

Valeur	Comportement
`auto` (défaut)	IPv6 si l'IPv6 est activé sur la sonde (chapitre 7) **et** si la cible possède une adresse AAAA ; IPv4 dans tous les autres cas
`ipv4`	IPv4 uniquement
`ipv6`	IPv6 uniquement. L'IPv6 doit être activé sur la sonde

> **Information** — La sonde fonctionne toujours en IPv4. L'IPv6 est facultatif et **désactivé par défaut** : il s'active dans la section `network` des paramètres (objet `ipv6`, chapitre 7). Le comportement de chaque type de contrôle vis-à-vis d'IPv6 est précisé aux chapitres 10 et 11 (§ 10.1.3).

L'identifiant doit commencer par une lettre minuscule ou un chiffre et ne contenir que des lettres minuscules, des chiffres et des tirets (2 à 63 caractères). Choisissez-le avec soin : il sert de clé dans l'historique, les tableaux de bord et les alertes. Le renommer revient à créer un nouveau contrôle du point de vue des outils d'observabilité.

Les durées s'écrivent sous la forme d'un nombre suivi d'une unité : `500ms`, `30s`, `5m`, `1h`.

> ✓ **Conseil** — Adoptez une convention de nommage dès le premier contrôle, par exemple `<application>-<fonction>` (`boutique-connexion`, `boutique-panier`, `erp-api-stock`). Les recherches, les filtres et les alertes en seront grandement simplifiés.

8.1.4 Exemple complet

L'exemple suivant, fictif, vérifie toutes les minutes que la page d'accueil d'un portail répond avec le code HTTP 200 et contient le texte attendu :

```json
{
  "id": "portail-accueil",
  "name": "Portail client : page d'accueil",
  "type": "http",
  "enabled": true,
  "tags": { "application": "portail", "environment": "production" },
  "schedule": { "interval": "1m", "jitter": "10s" },
  "timeout": "15s",
  "retry": { "attempts": 1, "interval": "20s" },
  "thresholds": { "warningDuration": "3s", "criticalDuration": "10s" },
  "http": {
    "method": "GET",
    "url": "https://portail.example.com/",
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "body", "operator": "contains", "value": "Espace client" }
    ]
  }
}
```

Le chapitre 9 explique comment créer ce contrôle par l'interface d'administration ou par l'API, et le chapitre 10 détaille chacune des options du type `http`.

8.2 Les cibles supervisées

8.2.1 Nature des cibles

La **cible** d'un contrôle est la ressource vérifiée : une adresse IP, un nom d'hôte, un port, une URL, un nom DNS, une base de données ou une application web. La sonde exécute ses contrôles **depuis le réseau où elle est installée**. Elle observe donc les ressources telles qu'un utilisateur ou un système situé au même endroit les perçoit, avec le même routage, les mêmes règles de pare-feu, le même serveur proxy et les mêmes serveurs DNS.

Cette position est déterminante pour l'interprétation des résultats :

- une ressource interne (`erp.example.internal`) n'est vérifiable que si la sonde est raccordée à un réseau qui y donne accès ;
- une ressource publique (`www.example.com`) est vérifiée **telle qu'elle est vue depuis votre site**, y compris à travers votre accès Internet ;
- deux sondes placées sur deux sites différents peuvent observer des résultats différents pour une même cible. C'est précisément ce qui permet de distinguer une panne générale d'un problème local.

8.2.2 Accès réseau aux cibles

Pour chaque cible, vérifiez avant de créer le contrôle que :

1. le nom de la cible est résolu par les serveurs DNS configurés sur la sonde (chapitre 7) ;
2. les flux nécessaires sont autorisés depuis l'adresse de la sonde vers la cible (port, protocole) ;
3. un passage par le proxy est configuré si la cible n'est joignable que par lui (chapitre 7) ;
4. les certificats présentés par la cible sont reconnus. Si elle utilise un certificat émis par une autorité de certification interne, importez cette autorité (chapitre 7).

> ! **Attention** — Une cible inaccessible pour des raisons de configuration réseau (flux non ouvert, proxy absent) produit des échecs qui ne reflètent pas l'état réel de l'application. Validez chaque nouveau contrôle par une exécution d'essai (chapitre 9, § 9.6.5) avant de l'activer.

8.2.3 Respect des cibles

La sonde génère du trafic réel vers les cibles. Ce trafic doit rester raisonnable et prévu :

- informez les équipes responsables des applications vérifiées, en particulier lorsqu'un contrôle utilise un compte de test ou s'authentifie ;
- évitez des fréquences inutilement élevées (§ 8.3.3) ;
- certaines cibles disposent de protections anti-robot ou de limitations de débit. Un contrôle trop fréquent peut les déclencher et faire bloquer l'adresse de la sonde ;
- les contrôles qui modifient des données (création de panier, commande de test) doivent viser un environnement de test ou des données de test identifiées (chapitre 15).

8.3 La fréquence d'exécution

8.3.1 Intervalle ou expression cron

La planification d'un contrôle s'exprime de l'une des deux façons suivantes, **jamais les deux à la fois** :

Mode	Syntaxe	Comportement	Usage typique
Intervalle	`{ "interval": "5m" }`	Une exécution toutes les 5 minutes, de façon continue	Disponibilité, temps de réponse
Expression cron	`{ "cron": "*/15 8-20 * * 1-5", "timezone": "Europe/Paris" }`	Exécutions aux instants décrits par l'expression, dans le fuseau horaire indiqué	Contrôles liés à des horaires (heures ouvrées, traitements nocturnes)

Un contrôle peut aussi n'avoir **aucune planification propre** (`"schedule": null`). Il n'est alors exécuté que manuellement ou au sein d'un plan d'exécution (chapitre 12).

Le chapitre 12 détaille la syntaxe des expressions cron, la gestion des fuseaux horaires et le dimensionnement de la charge.

8.3.2 La gigue

La **gigue** (`jitter`) est un décalage aléatoire, compris entre zéro et la valeur indiquée, appliqué au démarrage de chaque exécution. Elle évite que tous les contrôles ayant la même fréquence démarrent exactement au même instant, ce qui provoquerait des pics de charge sur la sonde comme sur les cibles. Exemple : `"jitter": "10s"` sur un intervalle d'une minute.

La gigue est particulièrement utile après un redémarrage de la sonde : sans elle, tous les contrôles seraient lancés simultanément (§ 8.14.2).

8.3.3 Choisir une fréquence

La fréquence fixe le **délai maximal de détection** d'un incident. Avec un intervalle de 5 minutes et une nouvelle tentative, un incident survenu juste après une exécution réussie est confirmé au plus tard après environ 5 minutes, plus l'intervalle de nouvelle tentative et la durée des exécutions.

Type de contrôle	Fréquence courante	Justification
`icmp`, `tcp`, `dns`	1 minute	Contrôles très légers ; détection rapide
`http`	1 à 5 minutes	Bon compromis entre réactivité et charge
`nagios`	Selon le plugin, souvent 1 à 5 minutes	Dépend du coût du plugin
`api`	2 à 10 minutes	Plusieurs requêtes par exécution
`database`	5 à 15 minutes, ou cron	Requêtes parfois coûteuses ; indicateurs métier à rythme horaire
`browser`	5 à 15 minutes	Exécution coûteuse en mémoire et en temps ; une seule à la fois par défaut

> **Information** — Une fréquence plus élevée améliore la réactivité, mais augmente le volume de données conservées (§ 8.12), la charge de la sonde et le trafic vers les cibles. Pour un parcours navigateur, une exécution toutes les 5 minutes représente déjà 288 parcours par jour.

8.4 Le délai maximal d'exécution (timeout)

Le **timeout** est la durée au-delà de laquelle une exécution est interrompue et considérée en échec. Il protège la sonde contre les exécutions bloquées et garantit que chaque contrôle produit un résultat dans un délai connu.

Quand le timeout est atteint :

1. la sonde interrompt l'exécution. Pour un plugin ou un scénario navigateur, elle met fin à l'ensemble des processus créés, y compris les processus secondaires de Chromium ;
2. l'exécution est enregistrée avec l'état `CRITICAL` et le type d'erreur `timeout` ;
3. pour un scénario navigateur, une capture d'écran de l'état de la page est produite lorsque c'est encore possible (chapitre 17).

Règles de réglage :

Règle	Explication
Le timeout doit dépasser nettement la durée normale	Prévoyez au moins deux à trois fois la durée habituelle, pour absorber les variations normales sans fausse alerte
Il doit rester inférieur à l'intervalle	Une exécution qui dure plus longtemps que l'intervalle retarde les suivantes (chapitre 12, § 12.6)
Il doit tenir compte de la plateforme	Le Raspberry Pi 5 est moins rapide qu'un poste de travail récent, notamment pour le lancement de Chromium et le rendu des pages

Règle	Explication
Les seuils de performance sont distincts	Le timeout signale une exécution qui n'aboutit pas ; la lenteur se signale avec `thresholds` (§ 8.6)

Type	Timeout indicatif
`icmp`, `tcp`, `dns`	5 à 10 s
`http`	10 à 30 s
`api`	30 à 60 s
`database`	30 à 60 s
`nagios`	10 à 60 s, selon le plugin
`browser`	60 à 180 s, selon le nombre d'étapes

Ces valeurs sont des points de départ. Ajustez-les après quelques jours d'observation des durées réelles (chapitre 17).

8.5 Nouvelles tentatives et confirmation d'un incident

8.5.1 Principe

Un échec isolé n'est pas toujours le signe d'un incident : une perte de paquet, un délai réseau ponctuel ou un redémarrage applicatif de quelques secondes peuvent faire échouer une exécution unique. Pour éviter les fausses alertes, la sonde peut **confirmer un échec** par une ou plusieurs nouvelles tentatives avant de le publier.

```json
"retry": { "attempts": 2, "interval": "30s" }
```

Avec ce réglage, un échec déclenche jusqu'à deux nouvelles exécutions, espacées de 30 secondes :

- si l'une d'elles réussit, l'échec n'est pas publié ;
- si toutes échouent, l'échec est confirmé et l'état `CRITICAL` est publié.

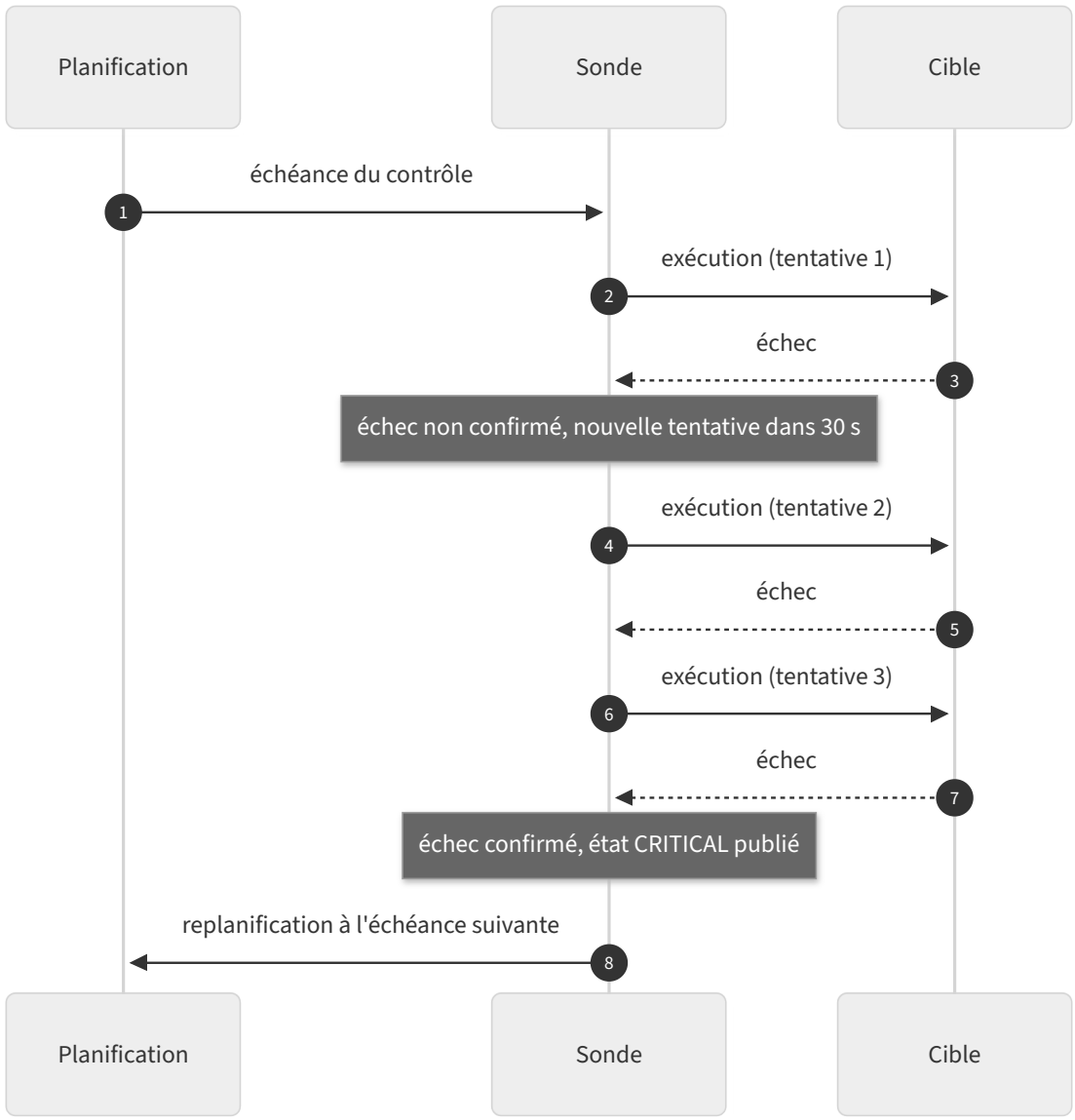


Schéma 8-2 — Nouvelles tentatives avant confirmation d'un échec

8.5.2 Valeurs et effets

Propriété	Valeurs	Défaut
`attempts`	0 à 5 nouvelles tentatives	0 (aucune)
`interval`	Durée entre deux tentatives	`15s`

Chaque tentative est enregistrée comme une exécution à part entière, avec son numéro de tentative (`attempt`). Elles restent toutes consultables dans l'historique, ce qui permet de repérer les instabilités intermittentes même lorsqu'elles n'ont pas abouti à une alerte.

> ✓ **Conseil** — Une nouvelle tentative suffit dans la plupart des cas. Au-delà de deux, le délai de détection s'allonge sensiblement pour un bénéfice faible. Pour les scénarios navigateur, tenez compte de la durée du scénario : trois tentatives d'un parcours de 60 secondes retardent la confirmation de plusieurs minutes et occupent le moteur de scénarios pendant ce temps.

Budget d'exécution. Dans le pire des cas, toutes les tentatives atteignent le timeout. Une exécution complète, nouvelles tentatives comprises, peut alors durer :

`(attempts + 1) × timeout + attempts × interval`

Par exemple, `"timeout": "90s"` et `"retry": { "attempts": 2, "interval": "30s" }` donnent 3 × 90 + 2 × 30 = 330 secondes, soit 5 min 30 s. Ce budget doit rester inférieur à l'intervalle de planification du contrôle, ou à la durée maximale du plan qui le contient ; sinon, des échéances sont sautées (chapitre 12, § 12.2.1 et § 12.6).

Parcours à effet de bord et idempotence. Une opération est **idempotente** lorsque la répéter produit le même effet que l'exécuter une seule fois : une consultation l'est, une création de commande ne l'est pas. Une nouvelle tentative **rejoue l'intégralité** du contrôle. Or une exécution interrompue de façon ambiguë (timeout, coupure réseau, interruption par l'activation d'une mise à jour avec le type d'erreur `maintenance`, § 8.14.2) a pu produire son effet sur l'application avant d'être interrompue : la sonde ne peut pas le savoir. La nouvelle tentative, ou l'exécution suivante, peut alors **rejouer cet effet** : deuxième commande de test, deuxième réservation, deuxième message.

Pour un scénario transactionnel, même en environnement de recette :

Recommandation	Raison
Fixer `"retry": { "attempts": 0 }`	Aucune répétition automatique d'une opération qui n'est pas idempotente
À défaut, rendre l'opération idempotente côté application, avec une **clé d'opération** préfixée pour signaler un objet de test et **réutilisée telle quelle à chaque répétition de la même opération logique** (même échéance de planification, nouvelles tentatives comprises), l'application refusant une seconde création avec la même clé	L'application ne reconnaît un doublon que si la clé est identique d'une tentative à l'autre : une référence tirée au hasard ou horodatée **à chaque exécution** change à chaque tentative et ne déduplique rien ; elle rend seulement chaque objet créé identifiable et rattachable à l'exécution qui l'a produit
Confier le **nettoyage** des objets de test à un mécanisme propre à l'environnement de recette (tâche planifiée de purge des objets marqués comme objets de test, réinitialisation périodique), indépendant de l'exécution suivante du scénario	Une étape de nettoyage placée à la fin du scénario ne s'exécute pas si celui-ci est interrompu avant son terme
Ne jamais exécuter un tel parcours en production sans l'accord du responsable de l'application	Voir chapitre 11, § 11.9.1

Le chapitre 15 (§ 15.8) présente un parcours de commande en environnement de test, auquel ces règles s'appliquent.

8.5.3 Bagotage

On parle de **bagotage** (*flapping*) lorsqu'un contrôle alterne rapidement entre réussite et échec. C'est souvent le symptôme d'un problème réel mais intermittent : saturation, équilibrage de charge vers un serveur défaillant, instabilité réseau. Les nouvelles tentatives masquent une partie de ces alternances. L'historique des exécutions et des tentatives permet de les mettre en évidence (chapitre 17).

8.6 Les seuils de performance

Une exécution peut réussir fonctionnellement tout en étant anormalement lente. Les **seuils de performance** (`thresholds`) permettent de qualifier cette lenteur :

Seuil	Effet si la durée totale de l'exécution le dépasse
`warningDuration`	L'exécution réussie passe à l'état `WARNING`
`criticalDuration`	L'exécution réussie passe à l'état `CRITICAL`

Exemple : avec `"warningDuration": "3s"` et `"criticalDuration": "10s"`, une page qui répond correctement en 4 secondes produit l'état `WARNING`, et en 12 secondes l'état `CRITICAL`. Au-delà du timeout, l'exécution est interrompue (§ 8.4).

Pour un contrôle `http` ou `api`, une assertion de type `duration` permet également de fixer une limite sur la durée d'une requête particulière (chapitres 10 et 11). Pour un scénario navigateur, la durée de chaque étape est mesurée et consultable, ce qui permet d'identifier l'étape responsable d'une lenteur (chapitre 17).

> ✓ **Conseil** — Fixez les seuils à partir des durées réellement observées sur une période représentative (au moins une semaine, incluant les heures de pointe), et non à partir d'objectifs théoriques. Un seuil trop bas génère des alertes `WARNING` permanentes qui finissent par être ignorées.

8.7 Le résultat d'une exécution

8.7.1 Contenu d'un résultat

Chaque exécution produit un résultat normalisé, identique dans sa structure pour tous les types de contrôles. Il est stocké sur la sonde, consultable dans l'interface d'administration et par l'API (`GET /api/v1/runs/{runId}`), et transmis aux destinations d'export configurées (chapitre 18).

Élément	Contenu	Exemple
Identifiant d'exécution	Identifiant unique, triable chronologiquement (ULID)	`01M7SHD1K2X7Q9N3V4T6W8Y0ZA`
Contrôle et révision	Identifiant du contrôle et numéro de version de sa configuration au moment de l'exécution	`boutique-connexion`, révision 7
Déclenchement	`schedule` (planification), `plan` (plan d'exécution), `manual`, `retry` (nouvelle tentative), `dryRun` (essai)	`retry`
Tentative	Numéro de tentative (§ 8.5)	`2`
Début et durée	Horodatage UTC et durée en millisecondes	`2026-11-19T09:15:02Z`, 14 650 ms
État	`OK`, `WARNING`, `CRITICAL` ou `UNKNOWN`	`CRITICAL`
Erreur	Type d'erreur normalisé (§ 8.8.5), étape concernée, message explicatif	`assertion_failed`, étape « Vérifier l'accès à l'espace client »
Étapes	Pour les scénarios et suites d'API : nom, état et durée de chaque étape	5 étapes
Mesures	Valeurs propres au type : code HTTP, phases de la requête, latence ICMP, valeur d'un indicateur métier, données de performance d'un plugin	`http.status_code = 200`

Élément	Contenu	Exemple
Sortie	Texte produit par un plugin Nagios	`HTTP OK: HTTP/1.1 200 OK - 512 bytes`
Artefacts	Fichiers de diagnostic : capture d'écran, journal de la console du navigateur, extrait réseau	`failure.png`
Trace	Identifiant de trace OpenTelemetry, pour retrouver l'exécution dans une plateforme d'observabilité	`4bf92f35…`

8.7.2 Lecture d'un résultat en échec

Le résultat d'un échec répond à trois questions, dans cet ordre :

1. **Quoi ?** L'état et le type d'erreur indiquent la nature du problème (délai dépassé, élément introuvable, code HTTP inattendu…).
2. **Où ?** L'étape en échec situe le problème dans le parcours.
3. **Pourquoi ?** Le message d'erreur, la capture d'écran et le journal de la console fournissent les éléments de diagnostic.

Exemple d'un résultat en échec, au format de l'API (fictif). Il s'agit de l'exécution du scénario déclaratif `boutique-connexion` (chapitre 14, § 14.3.1), repris à l'identique au chapitre 17 (§ 17.4) :

JSON

```
{
  "id": "01M7SHD1K2X7Q9N3V4T6W8Y0ZA",
  "checkId": "boutique-connexion",
  "checkRevision": 7,
  "planRunId": null,
  "trigger": "retry",
  "status": "CRITICAL",
  "attempt": 2,
  "startedAt": "2026-11-19T09:15:02.114Z",
  "durationMs": 14650,
  "error": {
    "type": "assertion_failed",
    "step": "Vérifier l'accès à l'espace client",
    "message": "Élément [data-testid=account-menu] non visible après 10000 ms"
  },
  "steps": [
    { "index": 0, "name": "Ouvrir la page de connexion", "status": "OK", "startedOffsetMs": 50,
    "durationMs": 1830 },
    { "index": 1, "name": "Saisir l'identifiant", "status": "OK", "startedOffsetMs": 1880,
    "durationMs": 160 },
    { "index": 2, "name": "Saisir le mot de passe", "status": "OK", "startedOffsetMs": 2040,
    "durationMs": 140 },
    { "index": 3, "name": "Valider le formulaire", "status": "OK", "startedOffsetMs": 2180,
    "durationMs": 2410 },
    { "index": 4, "name": "Vérifier l'accès à l'espace client", "status": "CRITICAL",
    "startedOffsetMs": 4590, "durationMs": 10004 }
  ],
  "measurements": { "browser.launch_ms": 1180 },
  "artifacts": [
    { "name": "failure.png", "kind": "screenshot", "bytes": 182344 },
    { "name": "console.json", "kind": "console", "bytes": 2210 }
  ],
  "trace": { "traceId": "4bf92f3577b34da6a3ce929d0e0e4736" }
}
```

Ce résultat se lit ainsi : il s'agit de la deuxième tentative (`attempt` vaut 2, `trigger` vaut `retry`), l'échec est donc confirmé. La page de connexion s'est ouverte, l'authentification a été soumise, mais l'espace client n'est pas apparu dans le délai de 10 secondes de l'étape, d'où la durée de cette étape et la durée totale de l'exécution. L'incident est donc probablement fonctionnel (authentification refusée ou page d'accueil du compte en erreur) plutôt que réseau. La capture d'écran le confirmera. Le chapitre 17 détaille la consultation et l'interprétation des résultats.

> **Information** — Les messages d'erreur sont produits avec **masquage des secrets** : si la valeur d'un secret apparaît dans un message, un journal ou une sortie de plugin, elle est remplacée avant tout enregistrement.

8.8 Les états et les erreurs

8.8.1 Les états

La sonde utilise cinq états. Ils reprennent la sémantique des plugins Nagios, familière aux équipes d'exploitation, et l'état `STALE`, qui signale l'absence de résultat récent.

État	Signification	Origine
`OK`	Le contrôle a réussi et respecte les seuils	Toutes les assertions et règles sont satisfaites, durée inférieure à `warningDuration` ; plugin retournant le code 0
`WARNING`	Le contrôle a réussi, mais avec une dégradation	Durée supérieure à `warningDuration` ; règle de sévérité `warning` non satisfaite (chapitre 11) ; certificat proche de l'expiration (chapitre 10) ; plugin retournant le code 1
`CRITICAL`	Le contrôle a échoué : la cible ne fonctionne pas comme attendu	Assertion ou règle non satisfaite, erreur réseau, timeout, durée supérieure à `criticalDuration` ; plugin retournant le code 2
`UNKNOWN`	Le contrôle n'a pas pu être évalué	Problème de la sonde ou de la configuration, et non de la cible : défaillance de Chromium, plugin introuvable, ressources insuffisantes, redémarrage pendant l'exécution, exécution interrompue par le drainage d'une activation de mise à jour (type d'erreur `maintenance`) ; plugin retournant le code 3
`STALE`	Aucun résultat récent n'est disponible	Contrôle actif sans exécution terminée depuis plus de **deux intervalles** de planification. Ce calcul est suspendu pendant l'état de maintenance de la sonde et pendant un intervalle après la reprise (§ 8.14.2)

Les quatre premiers états qualifient **une exécution**. `STALE` qualifie **un contrôle** : il est calculé à partir de l'absence d'exécution récente, et non produit par une exécution.

> **Attention** — `UNKNOWN` ne doit pas être traité comme un état anodin. Il signifie que la supervision ne fonctionne pas pour ce contrôle : un incident réel sur la cible pourrait passer inaperçu. Un `UNKNOWN` persistant appelle une intervention sur la sonde ou sur la configuration du contrôle (chapitre 19).

8.8.2 Règles de calcul de l'état

L'état d'une exécution est déterminé dans l'ordre suivant :

1. si l'exécution n'a pas pu être menée à son terme pour une raison propre à la sonde (moteur de scénarios indisponible, navigateur défaillant, mémoire insuffisante, redémarrage, interruption par le drainage d'une activation de mise à jour), l'état est `UNKNOWN` ;
2. sinon, si le timeout est atteint, si une erreur réseau ou protocolaire survient, si une assertion ou une règle de sévérité critique n'est pas satisfaite, ou si le code d'un scénario scripté lève une erreur (`script_error`), l'état est

`CRITICAL` ;

3. sinon, si la durée dépasse `criticalDuration`, l'état est `CRITICAL` ;
4. sinon, si la durée dépasse `warningDuration`, ou si une règle de sévérité `warning` n'est pas satisfaite, l'état est `WARNING` ;
5. sinon, l'état est `OK`.

Pour un contrôle `nagios`, le code de retour du plugin fixe directement l'état (0 → `OK`, 1 → `WARNING`, 2 → `CRITICAL`, 3 → `UNKNOWN`). Les seuils de durée de la sonde s'y ajoutent. Un code de retour en dehors de 0 à 3 est traité comme `UNKNOWN`, avec le type d'erreur `plugin_error`.

Une fois les nouvelles tentatives épuisées (§ 8.5), l'état de la dernière tentative devient **l'état courant du contrôle**. C'est cet état qui est publié vers les plateformes d'observabilité et renvoyé par `GET /api/v1/checks/{id}/status`.

8.8.3 Cycle de vie de l'état d'un contrôle

Outre les cinq états définis au § 8.8.1, le diagramme suivant fait apparaître deux **phases du cycle de vie** d'un contrôle, qui ne sont pas des états de résultat :

- **Inactif** : le contrôle est désactivé (`"enabled": false`). Il n'est pas planifié et n'est pas soumis au calcul de `STALE`, réservé aux contrôles actifs ; une exécution d'essai reste possible (chapitre 9, § 9.6.5) ;
- **En attente** : le contrôle est actif, mais aucune de ses exécutions n'est encore terminée, par exemple juste après sa création ou son activation. Dans la réponse de `GET /api/v1/checks/{id}/status`, le champ `lastRun` vaut alors `null`.

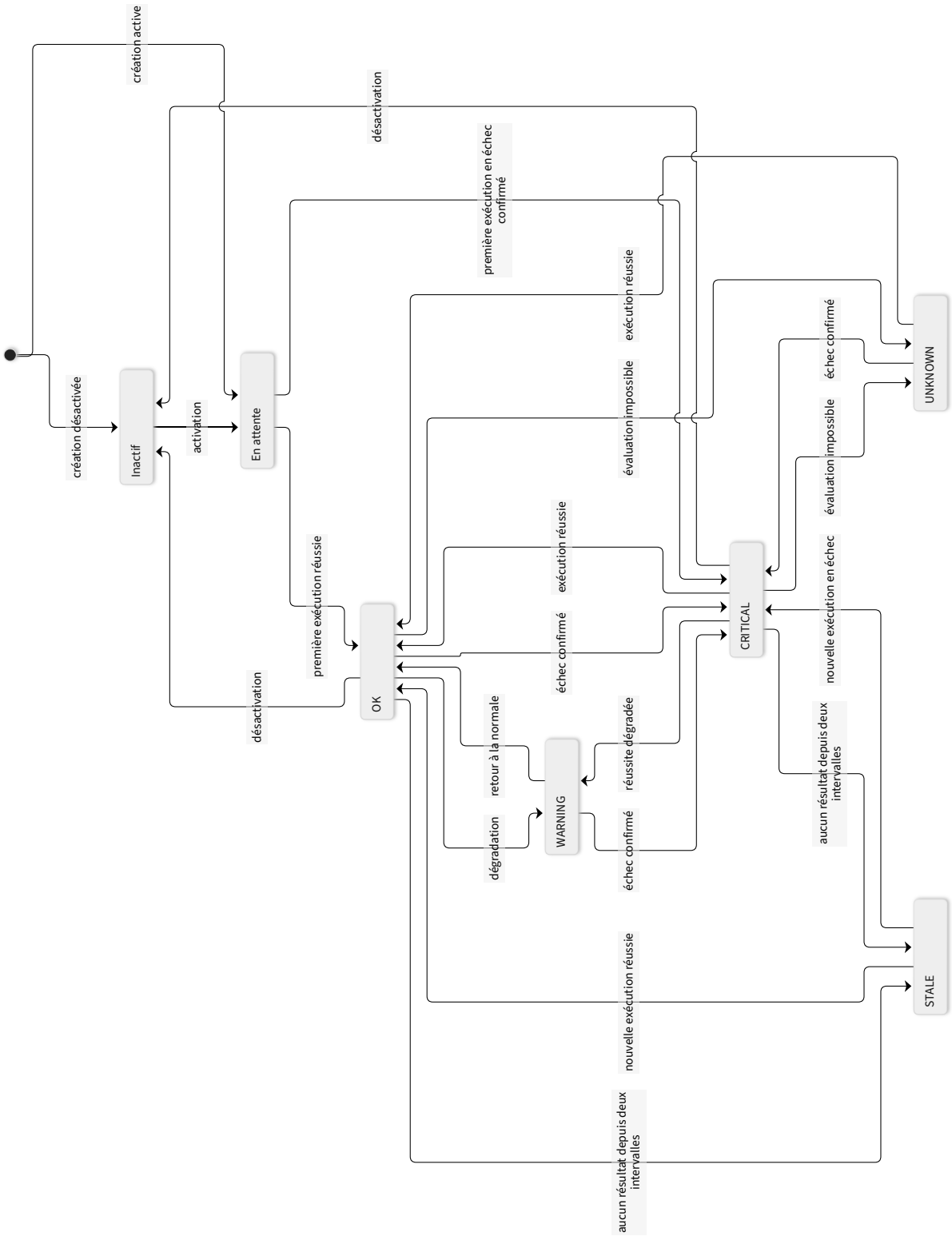


Schéma 8-3 — Cycle de vie de l'état d'un contrôle

Schéma présenté en orientation paysage : tournez le document d'un quart de tour pour le lire.

Pour la lisibilité, le diagramme ne représente pas toutes les transitions : tout état peut évoluer vers n'importe quel autre selon le résultat de l'exécution suivante, et tout contrôle peut être désactivé à tout moment. Chaque changement d'état d'un contrôle produit un événement `check.status.changed`, consultable par l'API (chapitre 9, § 9.11) et exporté sous forme de log (chapitre 18).

8.8.4 Le cas de l'état STALE

`STALE` peut apparaître à deux niveaux.

Niveau	Calculé par	Causes typiques
Contrôle	La sonde	File d'attente saturée (trop de contrôles pour les ressources disponibles), plan d'exécution bloquant les contrôles qu'il contient, moteur de scénarios arrêté
Sonde entière	La plateforme d'observabilité, par une alerte sur absence de signal	Sonde éteinte, débranchée, en panne, ou privée de réseau ; redémarrage lors de l'activation d'une mise à jour

La sonde ne peut évidemment pas signaler elle-même sa propre disparition. La détection d'une sonde muette repose donc sur une alerte configurée dans la plateforme d'observabilité : le chapitre 18 fournit les règles correspondantes pour chaque plateforme.

État de maintenance de la sonde. Outre son fonctionnement normal, la sonde peut se trouver en **état de maintenance** : elle y entre au début de l'activation d'une mise à jour (§ 8.14.2) et en sort après son redémarrage sur la nouvelle version. Cet état concerne la sonde entière, et non un contrôle. Il est exposé par `/health`, qui répond `{"status": "MAINTENANCE"}` avec le code HTTP 503, par la métrique `sva.agent.maintenance` (Prometheus : `sva_agent_maintenance`), par les événements OTLP `maintenance.started` et `maintenance.ended`, et par `check_sva`, qui renvoie le code 3 (`UNKNOWN`) avec le texte `MAINTENANCE` (chapitres 17 et 18). Pendant la maintenance et pendant un intervalle après la reprise, la sonde **suspend le calcul de `STALE`** pour ses contrôles : l'absence d'exécution due à l'activation ne produit pas d'état `STALE`. La durée de cet intervalle est un intervalle de planification de chaque contrôle après la reprise.

8.8.5 Les types d'erreur

Lorsqu'une exécution échoue (hors simple dépassement d'un seuil de durée ou de temps de réponse), le résultat précise un **type d'erreur** choisi dans une liste fermée (champ `error.type`). Cette liste permet des statistiques et des alertes fiables, ce que ne permettent pas des messages libres.

Le type d'erreur détermine l'état produit, selon la distinction présentée au § 8.10 :

- **`CRITICAL`** pour les erreurs qui mettent en cause la cible ou le parcours vérifié : `timeout`, `dns_error`, `connection_refused`, `connection_reset`, `tls_error`, `certificate_expired`, `http_status`, `assertion_failed`, `element_not_found`, `navigation_error`, `database_error`, `auth_failed`, ainsi que `script_error`, qui signale un parcours non vérifié ;
- **`UNKNOWN`** pour les erreurs qui empêchent l'évaluation : `browser_crash`, `plugin_error` (plugin qui n'a pas pu s'exécuter jusqu'à son terme ou dont le code retour est en dehors de 0 à 3, § 8.8.2), `secret_missing` (secret référencé par le contrôle mais absent de la sonde), `resource_exhausted`, `runner_crash`, `agent_restart`, `maintenance` (§ 8.14.2), `unknown`, et `no_data` par défaut (paramètre `onEmpty`, chapitre 11).

Le tableau de référence de tous les types d'erreur, avec leur signification, leur origine probable et la première vérification à mener, figure au chapitre 17 (§ 17.5.2). Le chapitre 19 les reprend sous forme de fiches de diagnostic.

8.9 Mesures de performance

8.9.1 Ce qui est mesuré

Chaque exécution mesure au minimum sa **durée totale**. Selon le type, la sonde collecte des mesures complémentaires qui permettent de localiser l'origine d'une lenteur.

Type	Mesures principales
`icmp`	Temps aller-retour (RTT), taux de perte de paquets
`tcp`	Durée d'établissement de la connexion
`dns`	Durée de résolution
`http`	Durée par phase : résolution DNS, connexion TCP, négociation TLS, délai avant le premier octet, transfert ; code HTTP ; durée de validité restante du certificat
`api`	Durée et code de chaque requête de la suite
`database`	Durée de la requête ; valeurs des indicateurs exportés
`nagios`	Données de performance (*perfdata*) publiées par le plugin
`browser`	Durée de chaque étape du parcours

8.9.2 Interpréter une durée

La durée mesurée par la sonde est celle observée **depuis son emplacement réseau**, avec ses propres ressources. Elle diffère de celle mesurée par un utilisateur sur un poste de travail, en particulier pour un parcours navigateur : le Raspberry Pi 5 exécute JavaScript et effectue le rendu des pages moins vite qu'un ordinateur récent.

Ce qui compte n'est donc pas la valeur absolue, mais **l'évolution** de la mesure dans le temps, pour une même sonde et un même contrôle. Une augmentation durable de 30 % du temps d'une étape est un signal fiable, même si la valeur absolue n'est pas directement comparable à l'expérience d'un utilisateur.

> ✓ **Conseil** — Pour comparer des sites entre eux, utilisez des sondes de même édition et de même configuration matérielle, avec des contrôles identiques.

8.10 Erreurs d'exécution et échecs de la cible

La distinction entre un problème **de la cible** et un problème **de la supervision** est fondamentale. Elle conditionne l'équipe à alerter et l'action à mener.

	Échec de la cible	Erreur d'exécution
Ce qui est en cause	L'application, le service ou le réseau supervisé	La sonde, sa configuration ou le contrôle lui-même
État produit	`CRITICAL` (ou `WARNING`)	`UNKNOWN`
Exemples	Code HTTP 500, élément absent de la page, base de données injoignable, valeur métier hors seuil	Chromium défaillant, mémoire insuffisante, plugin introuvable
Équipe concernée	Exploitation ou responsable de l'application	Administrateur de la sonde

	Échec de la cible	Erreur d'exécution
Action	Diagnostiquer l'application	Corriger la configuration ou intervenir sur la sonde (chapitre 19)

Certaines situations se situent à la frontière. Une erreur de programmation dans un scénario scripté (`script_error`) relève de la supervision, mais produit volontairement l'état `CRITICAL` : le parcours n'a pas été vérifié et ne doit pas passer inaperçu. De même, un contrôle dont le sélecteur ne correspond plus à la page, parce que l'application a évolué, produit un `CRITICAL` (`element_not_found`), alors que l'application fonctionne peut-être correctement. Ces **faux positifs** sont inhérents à la supervision synthétique. Le chapitre 16 décrit les pratiques qui les limitent (sélecteurs robustes, assertions explicites), et le chapitre 19 la manière de les diagnostiquer.

8.11 Classes d'exécution et concurrence

8.11.1 Principe

La sonde exécute plusieurs contrôles en parallèle, dans les limites de ses ressources. Pour qu'un contrôle coûteux ne bloque pas les contrôles légers, chaque type est rattaché à une **classe d'exécution** dotée de sa propre limite de **concurrence**, c'est-à-dire du nombre maximal d'exécutions simultanées.

Classe	Types de contrôles	Concurrence par défaut	Composant
Légère	`icmp`, `tcp`, `dns`, `http`, `api`	16	Moteur principal (ICMP : environnement isolé)
Plugins et bases	`nagios`, `database`	4	Environnement isolé (plugins) et moteur principal (bases)
Navigateur	`browser`	**1** (2 configurable sur une sonde de 8 Go)	Moteur de scénarios

Les valeurs de concurrence sont données à titre indicatif ; les valeurs en vigueur pour chaque version sont indiquées dans les notes de version.

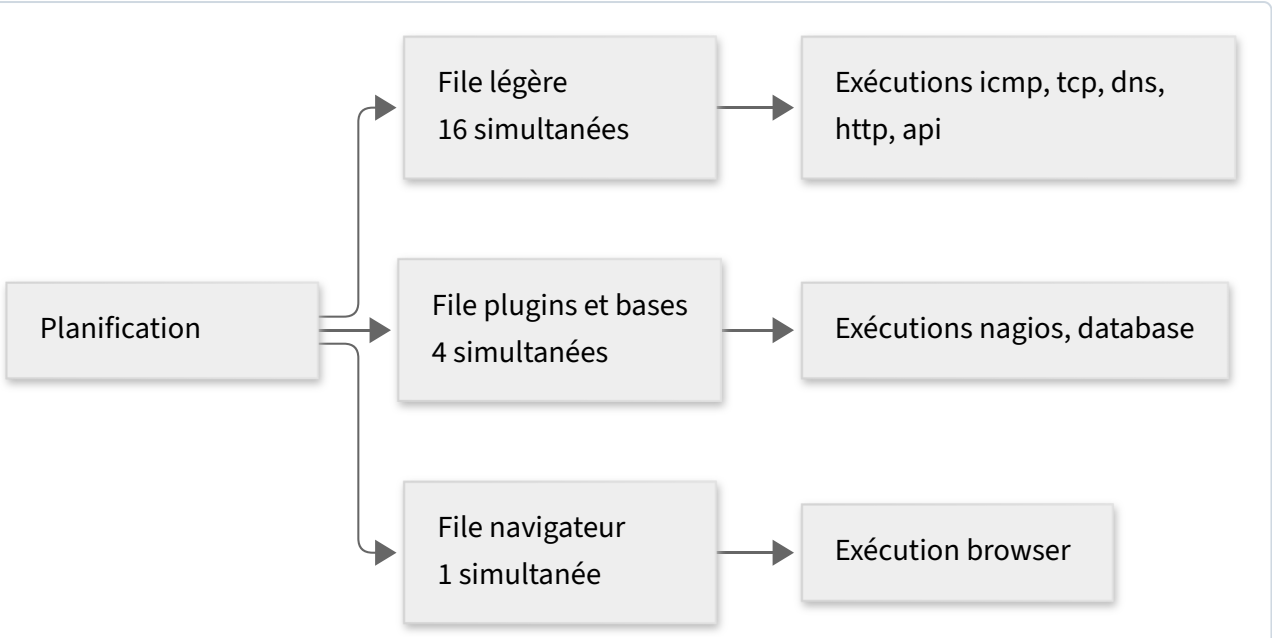


Schéma 8-4 — Files d'exécution et concurrence

8.11.2 Effets de la concurrence

Lorsqu'une file est pleine, les exécutions suivantes **attendent** qu'une place se libère. Ce retard est mesuré : c'est le **retard de planification**, publié dans les métriques de santé de la sonde (chapitre 18).

Un retard de planification durable indique que la sonde est sous-dimensionnée pour la charge configurée. C'est fréquent dans la classe navigateur, limitée à une exécution simultanée. Le chapitre 12 (§ 12.6) explique comment calculer la charge d'une sonde et la ramener dans des limites saines.

> **Attention** — Porter la concurrence de la classe navigateur au-delà de 2, ou à 2 sans mesure préalable de la consommation de vos scénarios, expose à des défaillances de Chromium par manque de mémoire, qui produisent des états `UNKNOWN` (`browser_crash`, `resource_exhausted`). Préférez réduire la fréquence des scénarios ou les regrouper dans un plan d'exécution.

8.12 Rétention et exploitation des données

8.12.1 Données conservées sur la sonde

La sonde conserve localement l'historique de ses exécutions. Elle reste ainsi exploitable sans aucune plateforme externe.

Donnée	Durée de conservation par défaut	Usage
Exécutions détaillées (résultat, étapes, mesures, erreurs)	14 jours	Diagnostic d'incidents récents
Agrégats horaires (nombre d'exécutions, d'échecs, durées médiane et 95e centile)	400 jours	Disponibilité et tendances sur l'année
Artefacts (captures d'écran, journaux de console)	7 jours, dans la limite de 500 Mo	Diagnostic visuel des échecs
Journal d'audit des opérations d'administration	365 jours	Traçabilité (chapitre 9)

Ces durées sont configurables par un administrateur (chapitre 9, § 9.9), dans la limite de l'espace de stockage disponible. Lorsque l'espace se raréfie, la sonde supprime d'abord les données les plus anciennes, en commençant par les artefacts, et poursuit l'exécution des contrôles.

> **Attention** — La rétention ne concerne que les données conservées **sur la sonde**. Les objets créés par un scénario dans l'application supervisée (commandes, comptes, paniers de test) ne sont jamais supprimés par la sonde. Leur nettoyage relève de l'environnement de recette, indépendamment des exécutions des scénarios (§ 8.5.2).

> **Information** — Le **95e centile** (p95) d'une série de durées est la valeur en dessous de laquelle se situent 95 % des mesures. Il reflète l'expérience des exécutions les plus lentes sans être faussé par quelques valeurs extrêmes, contrairement au maximum.

8.12.2 Volume de données

À titre indicatif, 50 contrôles exécutés toutes les minutes produisent environ 72 000 exécutions par jour, soit de l'ordre de 300 Mo sur 14 jours. Cette volumétrie est compatible avec une carte microSD de 32 Go. Elle justifie toutefois une fréquence raisonnable et le recours aux agrégats pour l'analyse de long terme.

SSD NVMe facultatif. Un SSD NVMe, raccordé par la carte M.2 HAT+ officielle de Raspberry Pi, peut accueillir les données volumineuses : historique des exécutions, agrégats, artefacts, file d'export et journaux persistants. Il est surtout utile pour l'édition Enterprise et pour les rétentions longues ; la microSD suffit à un usage standard. La partition de démarrage, les deux emplacements système, le certificat d'édition, la configuration et les secrets restent sur la microSD. Si le SSD est absent ou défaillant, les contrôles continuent, l'historique local est conservé sur la microSD en mode réduit, et l'indicateur de santé `storage` passe en `CRITICAL`. Le matériel compatible et l'activation sont décrits aux chapitres 4 et 20 (§ 20.3.4).

8.12.3 Exploitation des données

Les données de la sonde s'exploitent de trois manières complémentaires :

Usage	Moyen	Chapitre
Diagnostic immédiat, consultation sans outil tiers	Interface d'administration de la sonde, API REST	9, 17
Tableaux de bord, alertes, historique de long terme, vue multi-sondes	Plateformes d'observabilité via OpenTelemetry ou Prometheus	18
Intégration aux outils de supervision existants	Plugin `check_sva` pour Nagios et Centreon	18

La sonde reste dans tous les cas la **source de référence** des résultats : les plateformes externes en reçoivent une copie.

8.13 Incident réseau, applicatif ou fonctionnel

Un même symptôme vu par un utilisateur (« le site ne marche pas ») peut provenir de causes de natures très différentes. En combinant plusieurs types de contrôles sur une même application, on oriente rapidement le diagnostic vers la bonne équipe.

Nature de l'incident	Définition	Contrôles qui le révèlent	Signature typique
Réseau	La cible n'est pas joignable ou l'est mal : routage, pare-feu, DNS, saturation de lien	`icmp`, `tcp`, `dns`	`dns_error`, `connection_refused`, `timeout` sur des contrôles légers ; perte de paquets, latence en hausse
Technique / applicatif	La cible est joignable, mais le service applicatif répond mal ou pas du tout	`http`, `api`, `database`, `nagios`	`http_status` (5xx), `tls_error`, `database_error`, durées en forte hausse ; contrôles réseau `OK`
Fonctionnel	Le service répond techniquement, mais ne rend pas le service attendu	Assertions de contenu, `api` avec validation, `database` avec règles métier, `browser`	`assertion_failed`, `element_not_found`, `auth_failed` ; codes HTTP 200 ; contrôles réseau et techniques `OK`

La lecture combinée des états oriente le diagnostic :

Ping (`icmp`)	Port (`tcp`)	Page (`http`)	Parcours (`browser`)	Orientation probable
`CRITICAL`	`CRITICAL`	`CRITICAL`	`CRITICAL`	Incident réseau, ou serveur arrêté
`OK`	`CRITICAL`	`CRITICAL`	`CRITICAL`	Service arrêté ou flux filtré
`OK`	`OK`	`CRITICAL` (`http_status` 5xx)	`CRITICAL`	Incident applicatif côté serveur
`OK`	`OK`	`OK`	`CRITICAL` (`assertion_failed`)	Incident fonctionnel : parcours cassé, alors que le site répond
`OK`	`OK`	`WARNING`	`WARNING`	Dégradation de performance

Ping (`icmp`)	Port (`tcp`)	Page (`http`)	Parcours (`browser`)	Orientation probable
`OK`	`OK`	`OK`	`UNKNOWN`	Problème de la sonde ou du scénario, pas de l'application

> **Information** — Certains équipements filtrent ICMP. Un contrôle `icmp` en échec sur une cible dont le port applicatif répond n'indique pas nécessairement un incident : vérifiez d'abord qu'ICMP est autorisé vers cette cible avant d'en faire un critère d'alerte.

> **Conseil** — Pour chaque application importante, combinez au moins un contrôle technique (`tcp` ou `http`) et un contrôle fonctionnel (assertion de contenu, `api` ou `browser`). Cette combinaison distingue d'emblée « l'application est arrêtée » de « l'application fonctionne mal ».

8.14 Fonctionnement autonome

8.14.1 Indépendance vis-à-vis des plateformes externes

La sonde est conçue pour fonctionner **sans dépendance externe**. Sa configuration, sa planification, l'exécution des contrôles et l'historique des résultats résident entièrement sur l'appareil. Elle n'envoie **aucune donnée à l'éditeur** : ses seuls échanges avec lui sont la vérification et le téléchargement des mises à jour, lorsque vous les activez (chapitre 20).

Situation	Comportement de la sonde
Plateforme d'observabilité injoignable	Les contrôles continuent normalement. L'envoi de télémétrie est retenté pendant une durée limitée, puis les données non transmises sont abandonnées et comptabilisées. L'historique complet reste consultable localement
Accès Internet coupé	Les contrôles des ressources internes continuent. Ceux qui visent des ressources externes passent logiquement en `CRITICAL` : c'est le reflet fidèle de la situation vue depuis le site
Serveur NTP injoignable	La sonde conserve l'heure courante tant qu'elle n'est pas redémarrée. Après un redémarrage sans source de temps, les horodatages peuvent être faux tant que l'heure n'est pas resynchronisée : le Raspberry Pi 5 ne conserve l'heure hors tension qu'avec une pile optionnelle
Console centrale (évolution prévue) injoignable	Sans objet actuellement. Avec la console centrale prévue, la sonde continuera avec la dernière configuration reçue

> **Attention** — Une sonde privée d'accès vers ses destinations d'export continue de superviser, mais les alertes configurées dans ces plateformes ne se déclenchent plus. Configurez dans chaque plateforme une alerte d'absence de signal (chapitre 18), afin d'être prévenu de toute interruption de la remontée des données.

> **Évolution prévue** — Une version ultérieure permettra de retransmettre, après le rétablissement d'une destination d'export, les traces et événements produits pendant l'interruption. Seules les données encore présentes dans la file d'envoi au moment du rétablissement sont transmises.

8.14.2 Comportement au redémarrage

Après une coupure électrique ou un redémarrage, la sonde reprend son activité **sans intervention**.

1. Le système démarre, puis les services de la sonde sont lancés automatiquement.
2. La sonde vérifie l'intégrité de sa base de données. En cas d'anomalie, elle restaure la dernière sauvegarde de configuration (chapitre 20).
3. Les exécutions interrompues par l'arrêt sont clôturées avec l'état `UNKNOWN` et le type d'erreur `agent_restart`.
4. La planification est recalculée. Les exécutions manquées pendant l'arrêt ne sont **pas rattrapées** : chaque contrôle reprend à sa prochaine échéance, avec une gigue qui évite un démarrage simultané de tous les contrôles.
5. Les envois vers les destinations d'export reprennent dès que le réseau est disponible.

Une coupure électrique peut entraîner la perte des tout derniers résultats enregistrés. La configuration, conservée avec un niveau de durabilité supérieur, n'est pas affectée tant que le support de stockage respecte les écritures synchrones ; une panne de la carte elle-même relève de la sauvegarde de la configuration (chapitre 20).

Mises à jour. Une mise à jour du système est téléchargée et installée sur l'emplacement système inactif **sans interrompre les contrôles**. Seule son **activation**, c'est-à-dire le redémarrage sur la nouvelle version, interrompt la supervision. Elle a lieu :

- dans l'une des fenêtres de maintenance définies par le client (section `updates`, paramètre `windows`, chapitre 20). Une activation automatique ordinaire est annoncée au moins `updates.notice` à l'avance (24 heures par défaut) ; à défaut, elle est reportée à la fenêtre suivante. Un correctif critique est annoncé au moins 15 minutes à l'avance ;
- au moment choisi par un administrateur qui l'applique (`POST /api/v1/updates/apply`), immédiatement après confirmation explicite et sans préavis, ou à une heure donnée par `scheduledAt`, avec le préavis qui en résulte ;
- pour un correctif critique avec `criticalPolicy: asap`, dès que possible (chapitre 20, § 20.4.11).

Aucune fenêtre n'est définie par défaut : sans fenêtre, aucune activation automatique ordinaire n'a lieu, la mise à jour reste en attente et l'état de santé `updates` passe en `WARNING` (`CRITICAL` pour un correctif critique) ; seul un correctif critique en mode `asap`, choisi explicitement, est activé sans fenêtre. La matrice complète figure au chapitre 20, § 20.4.2. Si la nouvelle version n'est pas saine, la sonde revient automatiquement à la version précédente, au prix d'un redémarrage supplémentaire.

Au début d'une activation, la sonde **draine** ses exécutions avant de redémarrer :

1. Elle passe en **état de maintenance** (§ 8.8.4) : `/health` répond `MAINTENANCE` (HTTP 503) et l'événement `maintenance.started` est émis.
2. Aucune exécution nouvelle ne démarre.
3. Les exécutions en cours disposent du délai `updates.drainTimeout` (5 minutes par défaut, 30 minutes au maximum) pour se terminer normalement.
4. Au-delà de ce délai, les exécutions encore en cours sont interrompues avec l'état `UNKNOWN` et le type d'erreur `maintenance` (§ 8.8.5).
5. La sonde redémarre sur la nouvelle version, puis reprend son activité selon les étapes ci-dessus ; l'événement `maintenance.ended` signale la fin de l'état de maintenance.

Le calcul de `STALE` est suspendu pendant la maintenance et pendant un intervalle après la reprise (§ 8.8.4). L'état de maintenance **ne suffit pas à supprimer les alertes** dans vos plateformes, car la sonde est injoignable pendant son redémarrage : alignez leurs périodes de maintenance sur vos fenêtres (chapitre 18). La durée d'indisponibilité d'une activation s'observe sur vos sondes entre les événements `maintenance.started` et `maintenance.ended` (chapitre 20, § 20.4.5).

> **Information** — Une sonde unique ne supervise pas pendant son propre redémarrage. Lorsque la continuité de la supervision est essentielle, déployez plusieurs sondes par site, avec des fenêtres de maintenance qui ne se chevauchent pas (chapitre 20).

> **Avertissement** — Des coupures électriques répétées sollicitent fortement la carte microSD et peuvent, à terme, l'endommager. Préférez toujours un arrêt propre (chapitre 20) et alimentez la sonde par une alimentation conforme (chapitre 4), idéalement secourue par onduleur.

8.14.3 Ce que la sonde ne fait pas seule

Pour fixer des attentes justes :

- la sonde **ne crée pas d'alertes** par courriel ou messagerie. Les alertes sont produites par les plateformes vers lesquelles elle exporte ses données, ou par Nagios/Centreon via `check_sva` (chapitre 18) ;
- la sonde **ne corrige pas** les incidents qu'elle détecte. Elle les constate et fournit les éléments de diagnostic ;
- la sonde **n'observe que ce qu'on lui demande** : un parcours non couvert par un contrôle n'est pas supervisé.

Chapitre 9 — Administration de la sonde

Ce chapitre décrit comment administrer la sonde au quotidien :

- les deux moyens d'administration : l'interface web et l'API REST ;
- la gestion des comptes, des jetons d'API et des secrets ;
- la création et le pilotage des contrôles ;
- les paramètres, le journal d'audit et la sauvegarde de la configuration ;
- l'automatisation de ces opérations.

Il suppose que la mise en service décrite au chapitre 6 est terminée : un compte administrateur existe, et la sonde est joignable en HTTPS depuis votre poste.

Les noms d'hôtes, adresses et identifiants des exemples sont fictifs. La sonde y est désignée par `sonde-paris-01.example.internal`, et l'adresse `192.0.2.50` est réservée à la documentation.

> **Information** — L'interface d'administration et l'API sont décrites ici par leurs fonctions. Les écrans de l'interface ne sont pas reproduits en détail : chaque opération est aussi décrite par la requête REST correspondante.

9.1 Moyens d'administration

9.1.1 Vue d'ensemble

La sonde s'administre **localement**, depuis le réseau où elle est installée, sans dépendre d'aucun service extérieur. Deux moyens sont disponibles, et ils offrent les mêmes possibilités.

Moyen	Adresse	Destiné à	Authentification
Interface d'administration	`https://<adresse-de-la-sonde>/`	Mise en service, consultation, opérations courantes	Identifiant et mot de passe d'un compte local
API REST	`https://<adresse-de-la-sonde>/api/v1`	Automatisation, intégration à vos outils, administration en masse	Jeton d'API

L'interface d'administration s'appuie elle-même exclusivement sur l'API REST. Toute opération réalisable dans l'interface l'est donc aussi par l'API, avec les mêmes règles de droits et de validation.

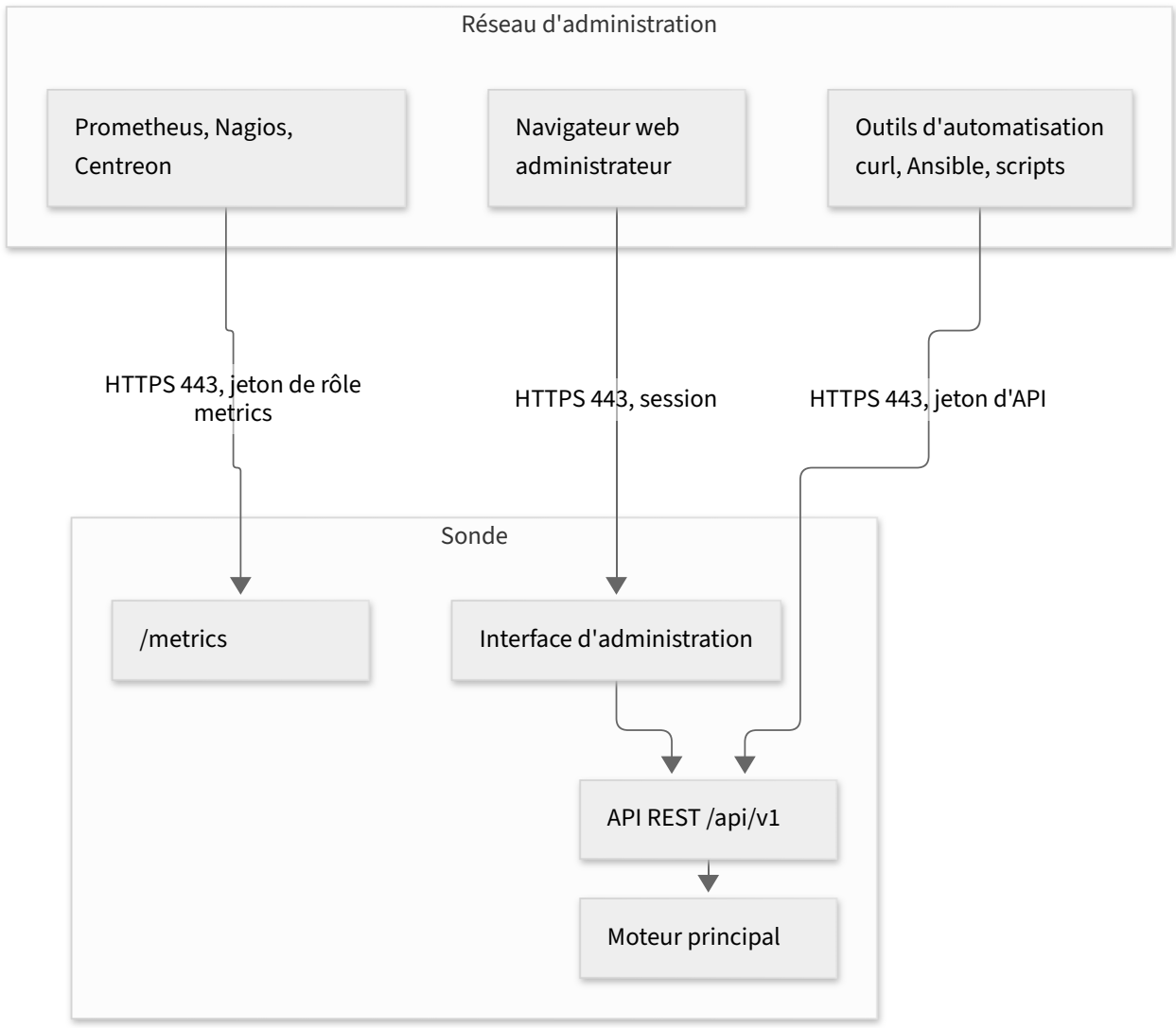


Schéma 9-1 — Interface d'administration et API REST

> **Avertissement** — L'interface d'administration et l'API sont conçues pour être utilisées depuis le réseau où la sonde est installée. N'exposez jamais la sonde directement sur Internet, par exemple via une redirection de port. L'administration à distance doit passer par un moyen d'accès sécurisé à ce réseau (accès d'administration, bastion, VPN), celui de votre organisation ou celui convenu avec l'exploitant du réseau d'accueil. Le chapitre 7 précise les flux à autoriser.

9.1.2 Accès SSH

L'accès SSH au système de la sonde est **désactivé par défaut**. Il n'est pas nécessaire à l'administration courante, qui se fait entièrement par l'interface et l'API. Un administrateur peut l'activer pour une opération de maintenance avancée ou à la demande du support (chapitre 7). Désactivez-le dès la fin de l'intervention.

L'activation est toujours **temporaire** : elle exige une date de fin (`enabledUntil`), fixée à 4 heures par défaut et à 72 heures au maximum. À l'échéance, SSH se désactive automatiquement et les sessions ouvertes sont fermées. Seul un compte `admin` de la sonde peut l'activer ; l'éditeur ne dispose d'aucun accès permanent. Activation, connexions et

désactivation sont enregistrées dans le journal d'audit (§ 9.10). Si votre politique de sécurité l'exige, SSH peut aussi être **verrouillé définitivement** (§ 9.9.3).

9.2 Interface d'administration

9.2.1 Accès

Objectif : ouvrir une session sur l'interface d'administration.

Prérequis : la sonde est mise en service (chapitre 6) ; vous disposez d'un compte et de l'empreinte du certificat de la sonde, ou le certificat de la sonde a été remplacé par un certificat de votre autorité de certification (chapitre 7).

Étapes :

1. Dans votre navigateur, saisissez l'adresse `https://<adresse-de-la-sonde>/`, par exemple `https://sonde-paris-01.example.internal/`.
2. Si la sonde utilise encore son certificat autosigné, le navigateur affiche un avertissement de sécurité. Consultez le certificat présenté et comparez son empreinte SHA-256 avec celle notée lors de la mise en service. Poursuivez uniquement si elles sont identiques.
3. Saisissez votre identifiant et votre mot de passe.

Résultat attendu : le tableau de bord des états des contrôles s'affiche.

> ! **Avertissement** — Un avertissement de certificat dont l'empreinte ne correspond pas à celle de la sonde peut signaler une interception du trafic. Dans ce cas, ne saisissez pas vos identifiants et alertez votre équipe sécurité. Pour supprimer durablement l'avertissement, installez sur la sonde un certificat émis par l'autorité de certification de l'entreprise (chapitre 7).

9.2.2 Fonctions disponibles

Domaine	Fonctions	Rôle minimal
Tableau de bord	État courant de tous les contrôles, filtres par type, état et étiquette	`viewer`
Résultats	Historique des exécutions, détail d'une exécution (étapes, erreur, mesures), captures d'écran et journaux	`viewer`
Contrôles	Création, modification, suppression, activation, désactivation, exécution immédiate et exécution d'essai	`operator`. Pour un contrôle scripté ou à plugin personnalisé : création, remplacement complet (`PUT`) et tout changement de code (`browser.mode`, `browser.scriptRef`, plugin) réservés à `admin` ; activation, désactivation, exécutions, réglages par `PATCH` et suppression restent ouverts à `operator`
Plans d'exécution	Création, modification, activation, désactivation, exécution immédiate	`operator`
Scripts	Téléversement des scénarios scriptés	`admin`
Secrets	Création, rotation, suppression (valeurs jamais affichées)	`admin`
Destinations d'export	Configuration et test des destinations OpenTelemetry et Prometheus	`admin`

Domaine	Fonctions	Rôle minimal
Paramètres	Réseau (dont IPv6 et 802.1X), heure, proxy, rétention, sécurité, SSH, mises à jour, licence, justificatif de maintenance, certificat, stockage SSD	`admin`
Comptes et jetons	Gestion des utilisateurs et des jetons d'API	`admin` (changement de son propre mot de passe : tout utilisateur authentifié)
Journal d'audit	Consultation des opérations d'administration	`admin`
Santé de la sonde	Version, édition, ressources, état des composants et des exports	`viewer`
Sauvegarde	Export et import de la configuration	`admin`

Les fonctions non autorisées pour votre rôle ne sont pas proposées.

9.2.3 Session

La session est maintenue par un cookie sécurisé, transmis uniquement en HTTPS et inaccessible aux scripts de la page. Elle expire après 30 minutes d'inactivité, et dans tous les cas 12 heures après la connexion. Pensez à vous déconnecter explicitement sur un poste partagé.

Après 5 tentatives de connexion infructueuses consécutives, le compte et l'adresse d'origine sont bloqués pendant 15 minutes. Toutes les tentatives, réussies ou non, sont enregistrées dans le journal d'audit (§ 9.10).

9.3 API REST : principes

9.3.1 Base et format

Élément	Règle
Adresse de base	`https://<adresse-de-la-sonde>/api/v1`
Version	Indiquée dans le chemin (`v1`). Une future version majeure coexistera au moins 12 mois avec la précédente
Format	JSON encodé en UTF-8 (`Content-Type: application/json`)
Dates	RFC 3339, en UTC. Exemple : `2025-11-19T09:15:02Z`
Durées dans les résultats	En millisecondes, dans des champs suffixés `Ms` (`durationMs`)
Durées dans la configuration	Chaînes `500ms`, `30s`, `5m`, `1h`
Identifiants	Choisis par vous pour les objets de configuration (`portail-accueil`) ; générés par la sonde (ULID) pour les exécutions
Description de l'API	Document OpenAPI 3.1 disponible à l'adresse `GET /api/v1/openapi.json`

Le document OpenAPI décrit toutes les ressources, leurs paramètres et leurs schémas. Il peut être importé dans un outil de test d'API ou servir à générer un client dans le langage de votre choix.

Deux adresses se situent hors de l'arborescence `/api/v1` :

Adresse	Usage	Authentification
`GET /health`	Vérifier que la sonde répond (vivacité). Ne renvoie aucune information sensible. Réponse `{"status": "OK"}` (code `200`) ; pendant l'**état de maintenance** qui ouvre l'activation d'une mise à jour, `{"status": "MAINTENANCE"}` (code `503`) (chapitre 20, § 20.4.9)	Aucune
`GET /metrics`	Métriques au format Prometheus (chapitre 18)	Jeton de rôle `metrics`

9.3.2 Authentification

Chaque appel à l'API, hors `GET /api/v1/system/certificate` (téléchargement du certificat de la sonde, sans authentification), doit présenter un **jeton d'API** dans l'en-tête HTTP `Authorization` :

```http
GET /api/v1/checks HTTP/1.1
Host: sonde-paris-01.example.internal
Authorization: Bearer <jeton>
Accept: application/json
```

Le jeton détermine l'identité de l'appelant et son rôle (§ 9.4). Un appel sans jeton, ou avec un jeton invalide, expiré ou révoqué, reçoit la réponse `401 Unauthorized`. Un appel authentifié mais non autorisé pour le rôle du jeton reçoit `403 Forbidden`.

> → **Évolution prévue** — L'authentification par certificat client (mTLS) n'est pas disponible actuellement : elle est prévue dans une version ultérieure, avec la console centrale (§ 9.14).

9.3.3 Préparer son poste : certificat et jeton

Les exemples de ce chapitre utilisent l'outil en ligne de commande `curl`, disponible sur Linux, macOS et Windows. Ils suivent deux règles de sécurité :

- **le certificat de la sonde est toujours vérifié**, grâce à l'option `--cacert`. L'option `-k` (`--insecure`) désactive cette vérification et ne doit pas être utilisée ;
- **le jeton n'apparaît jamais dans la ligne de commande**, où il serait visible dans l'historique du shell et la liste des processus. Il est lu depuis un fichier protégé.

Objectif : préparer un poste Linux ou macOS pour appeler l'API de façon sûre.

Prérequis : `curl` et `openssl` installés ; un jeton d'API (§ 9.5) ; l'empreinte SHA-256 du certificat de la sonde.

Étapes :

1. Créez un répertoire de travail accessible à vous seul :

   ```bash
   mkdir -p "$HOME/.sva" && chmod 700 "$HOME/.sva"
   ```

2. Si la sonde utilise son certificat autosigné, récupérez-le, soit par l'adresse `GET /api/v1/system/certificate` (chapitre 6, § 6.4.4), soit avec `openssl` :

   ```bash
   openssl s_client -connect sonde-paris-01.example.internal:443 \
     -servername sonde-paris-01.example.internal </dev/null 2>/dev/null \
     | openssl x509 -outform PEM > "$HOME/.sva/sonde-paris-01.pem"
   ```

3. Affichez son empreinte et **comparez-la** avec celle relevée lors de la mise en service :

```bash
openssl x509 -in "$HOME/.sva/sonde-paris-01.pem" -noout -fingerprint -sha256
```

   Si les empreintes diffèrent, supprimez le fichier et n'allez pas plus loin.

   Si la sonde utilise un certificat émis par l'autorité de certification de l'entreprise, utilisez à la place le certificat de cette autorité.

4. Créez le fichier d'en-tête d'authentification `entete-auth` une fois, puis recréez-le de la même manière à chaque rotation du jeton. Le répertoire `$HOME/.sva` est d'abord rendu privé ; un éventuel ancien fichier est supprimé, car un masque de permissions (`umask`) ne s'applique qu'à la création et ne resserre pas les droits d'un fichier existant ; le nouveau fichier est créé vide en mode `600` **avant** que le jeton n'y soit écrit. La commande `read -rs`, propre à bash et à zsh, lit le jeton au clavier sans l'afficher ; les parenthèses limitent le masque et la variable à cette seule opération :

```bash
(
  umask 077
  mkdir -p "$HOME/.sva" && chmod 700 "$HOME/.sva" &&
  rm -f "$HOME/.sva/entete-auth" &&
  : > "$HOME/.sva/entete-auth" && chmod 600 "$HOME/.sva/entete-auth" &&
  IFS= read -rs SONDE_TOKEN &&
  printf 'Authorization: Bearer %s\n' "$SONDE_TOKEN" > "$HOME/.sva/entete-auth"
)
```

   Si le jeton vous a été remis dans un fichier, par exemple `$HOME/.sva/jeton`, la forme suivante est équivalente (`printf` est une commande interne du shell : le jeton ne figure dans les arguments d'aucun processus) :

```bash
(
  umask 077
  mkdir -p "$HOME/.sva" && chmod 700 "$HOME/.sva" &&
  rm -f "$HOME/.sva/entete-auth" &&
  : > "$HOME/.sva/entete-auth" && chmod 600 "$HOME/.sva/entete-auth" &&
  printf 'Authorization: Bearer %s\n' "$(cat "$HOME/.sva/jeton")" > "$HOME/.sva/entete-auth"
)
```

   Supprimez ensuite le fichier `jeton` s'il n'a plus d'usage.

5. Définissez des variables de confort pour la session :

```bash
SONDE=https://sonde-paris-01.example.internal
CA="$HOME/.sva/sonde-paris-01.pem"
```

Information — Pourquoi cette méthode protège le jeton :

- le répertoire `$HOME/.sva` est ramené aux droits `700` et l'ancien fichier `entete-auth` éventuel est supprimé, car `umask` ne modifie pas les droits d'un fichier existant ; le nouveau fichier naît vide avec les droits `600` (lecture et écriture pour vous seul), et le jeton n'y est écrit qu'ensuite : il n'existe aucun instant où le jeton serait lisible par un autre compte. Chaque commande est enchaînée par `&&` : si l'une échoue, le jeton n'est pas écrit ;
- `read` et `printf` sont des **commandes internes** du shell (bash ou zsh ; l'option `-s` de `read` n'existe pas dans le shell POSIX `sh`) : elles ne créent aucun processus, et le jeton n'apparaît donc ni dans la liste des processus (`ps`), ni dans l'historique du shell. Dans la seconde forme, le processus `cat` ne reçoit que le **chemin** du fichier, jamais le jeton. Vous pouvez le vérifier avec `type printf`, qui doit répondre `printf is a shell builtin`. N'appelez pas `/usr/bin/printf` ou `env printf`, qui seraient des programmes externes ;
- `curl -H @"$HOME/.sva/entete-auth"` lit ensuite l'en-tête dans le fichier : le jeton ne figure pas dans les arguments de `curl`.

N'activez pas la trace du shell (`set -x`) pendant ces opérations : elle afficherait le jeton. Pour renouveler un jeton (§ 9.5), recréez simplement le fichier selon l'étape 4.

Vérification : interrogez l'état de la sonde.

```bash
curl --fail-with-body -sS --cacert "$CA" -H @"$HOME/.sva/entete-auth" \
  "$SONDE/api/v1/system"
```

Résultat attendu : une réponse JSON décrivant la sonde (version, édition, ressources, état des composants). Sa forme est définie par le schéma `system.schema.json`, et ses champs sont commentés au chapitre 17 (§ 17.9).

Information — `-H @fichier` demande à `curl` de lire les en-têtes dans un fichier. L'option `--fail-with-body` fait échouer la commande sur une erreur HTTP tout en affichant le détail renvoyé par la sonde. Ces options requièrent une version récente de `curl` : 7.55 ou ultérieure pour `-H @fichier`, 7.76 ou ultérieure pour `--fail-with-body`.

Dans la suite du chapitre, les commandes supposent ces variables définies. Pour alléger les exemples, nous utilisons la fonction suivante :

```bash
# Appel authentifié à l'API de la sonde, certificat vérifié
sapi() {
  curl --fail-with-body -sS --cacert "$CA" -H @"$HOME/.sva/entete-auth" \
       -H 'Accept: application/json' "$@"
}
```

L'outil `jq`, cité dans plusieurs exemples, met en forme et filtre les réponses JSON. Son usage est facultatif.

9.3.4 Codes de réponse et erreurs

Code	Signification	Cas typique
`200 OK`	Succès avec contenu	Lecture, modification
`201 Created`	Objet créé ; l'en-tête `Location` indique son adresse	Création d'un contrôle

Code	Signification	Cas typique
`202 Accepted`	Action acceptée et lancée en arrière-plan ; l'en-tête `Location` indique où suivre son avancement	Exécution manuelle, test d'une destination d'export
`204 No Content`	Succès sans contenu	Suppression, activation
`304 Not Modified`	Ressource inchangée depuis la version que vous détenez	Lecture conditionnelle
`400 Bad Request`	Requête mal formée	JSON invalide
`401 Unauthorized`	Jeton absent, invalide, expiré ou révoqué	—
`403 Forbidden`	Rôle insuffisant, fonction non incluse dans l'édition, ou mot de passe actuel erroné lors du changement de son propre mot de passe. Ces cas se distinguent par le champ `type` de l'erreur : `/api/v1/problems/edition-feature-unavailable` pour une fonction absente de l'édition, `/api/v1/problems/current-password-invalid` pour un mot de passe actuel erroné (§ 9.4.2)	Rôle `operator` qui téléverse un script ; contrôle `browser` créé sur une sonde Professional ; `PUT /api/v1/users/me/password` avec un mot de passe actuel erroné
`404 Not Found`	Objet inexistant	Identifiant erroné
`405 Method Not Allowed`	Méthode non admise par la ressource	`PUT /api/v1/settings/license`, section en lecture seule
`409 Conflict`	Opération incompatible avec l'état actuel	Création d'un identifiant déjà utilisé ; suppression d'un secret référencé par un contrôle
`410 Gone`	Ressource qui n'est plus disponible	Curseur de pagination expiré, plus de 24 heures après sa délivrance (§ 9.3.6)
`412 Precondition Failed`	Condition de l'en-tête `If-Match` ou `If-None-Match` non satisfaite (§ 9.3.5)	Objet modifié depuis votre dernière lecture ; création d'un secret qui existe déjà
`422 Unprocessable Content`	Contenu valide en JSON, mais non conforme au schéma ou aux règles	Champ obligatoire manquant, durée invalide
`428 Precondition Required`	Ni `If-Match` ni `If-None-Match` dans une requête qui en exige un (§ 9.3.5)	`PUT` ou `DELETE` sans en-tête conditionnel
`429 Too Many Requests`	Limite de débit atteinte	Boucle d'automatisation trop rapide
`5xx`	Erreur interne de la sonde	À signaler au support si elle persiste

Les erreurs sont décrites au format **Problem Details** (RFC 9457), type de contenu `application/problem+json`. Exemple de refus de validation (fictif) :

```json
{
  "type": "/api/v1/problems/validation",
  "title": "Configuration invalide",
  "status": 422,
  "detail": "Le contrôle 'portail-accueil' ne respecte pas le schéma.",
  "errors": [
    { "pointer": "/schedule", "detail": "Indiquer 'interval' ou 'cron', mais pas les deux." },
    { "pointer": "/timeout", "detail": "Format de durée invalide : '15 secondes'." }
  ]
}
```

Le champ `type` est une adresse relative de la forme `/api/v1/problems/{code}`, servie par la sonde. Il permet à un outil de distinguer les causes d'un même code HTTP : un refus `403` lié à l'édition porte ainsi le type `/api/v1/problems/edition-feature-unavailable`, et un refus pour mot de passe actuel erroné le type `/api/v1/problems/current-password-invalid`, et non celui d'un rôle insuffisant.

9.3.5 Modifications concurrentes : ETag et If-Match

Plusieurs administrateurs, ou un administrateur et un outil d'automatisation, peuvent modifier le même objet. Pour éviter qu'une modification n'en écrase silencieusement une autre, la sonde applique un **contrôle de concurrence optimiste** :

1. Chaque lecture d'un objet de configuration renvoie un en-tête `ETag`, qui identifie sa version. Pour un secret ou un compte, cette lecture est la lecture individuelle : `GET /api/v1/secrets/{name}` (§ 9.8.4) ou `GET /api/v1/users/{id}` (§ 9.4.2), réservées au rôle `admin`.
2. Toute modification (`PUT`, `PATCH`) ou suppression (`DELETE`) d'un objet existant doit présenter cette valeur dans l'en-tête `If-Match`.
3. Une ressource **créée** par `PUT`, comme un secret (`PUT /api/v1/secrets/{name}`, § 9.8.2), n'a pas encore d'`ETag` : la création présente l'en-tête `If-None-Match: *`, qui signifie « uniquement si la ressource n'existe pas ». Si elle existe déjà, la sonde répond `412 Precondition Failed` : il s'agit alors d'un remplacement, qui exige `If-Match`.
4. Une requête `PUT`, `PATCH` ou `DELETE` qui ne présente ni `If-Match` ni `If-None-Match` est refusée avec le code `428 Precondition Required`.
5. Si l'objet a changé entre-temps, la sonde refuse l'opération avec le code `412 Precondition Failed`. Relisez alors l'objet, réappliquez votre modification sur la version actuelle et renvoyez-la.

Portée. Cette règle s'applique aux ressources de configuration : contrôles, plans, scripts, destinations d'export, secrets, comptes et sections de paramètres (`/api/v1/settings/{section}`). Les requêtes d'action et d'import en écriture seule, qui ne modifient pas une ressource relue au préalable (`PUT /api/v1/system/maintenance`, `PUT /api/v1/system/certificate`, `PUT /api/v1/system/dot1x/credential`), n'en relèvent pas.

Exception. Le changement de son propre mot de passe, `PUT /api/v1/users/me/password` (§ 9.4.2), n'exige ni `If-Match` ni `If-None-Match` : la preuve demandée est le mot de passe actuel, et non la version du compte. Cette requête ne reçoit donc jamais le code `428`.

```bash
# Lecture : afficher l'ETag renvoyé
sapi -D - -o /dev/null "$SONDE/api/v1/checks/portail-accueil" | grep -i '^etag'
```

9.3.6 Pagination et filtres

Les listes potentiellement longues, comme l'historique des exécutions, sont **paginées par curseur**. La réponse contient les éléments et, s'il en reste, un curseur vers la page suivante :

```json
{
  "items": [ { "id": "01M7SJ4Q8R2V6X9Z3B5D7F1H0K", "checkId": "portail-accueil", "status": "OK" } ],
  "nextCursor": "eyJpZCI6IjAxSkRCNUs4UjJYN1E5TjNWNFQ2VzhZMFpBIn0"
}
```

Pour obtenir la page suivante, renvoyez la même requête avec le paramètre `cursor`. Le paramètre `limit` fixe la taille de page (100 par défaut, 1 000 au maximum). Le curseur est une valeur opaque, encodée en base64url : il se transmet tel quel, sans encodage supplémentaire.

Cohérence de la pagination. Pour l'historique des exécutions, les règles suivantes s'appliquent :

Règle	Conséquence pour vous
Les éléments sont rangés selon un **ordre total** : date de début (`startedAt`), puis identifiant (`id`) pour départager deux exécutions de même date	Aucun élément n'est lu deux fois ni sauté du fait d'un ex æquo
Le curseur encode la **dernière position lue** et reste valide **24 heures**	Au-delà, la sonde répond `410 Gone` : relancez la lecture depuis la première page
Les exécutions enregistrées pendant la lecture n'apparaissent dans les pages suivantes que si leur date de début les place après la dernière position lue ; sinon, elles manquent à la lecture en cours	Pour un **export reproductible**, fixez toujours une borne `to` dans le passé : les exécutions commencées après elle sont exclues. Une exécution commencée avant `to` mais enregistrée plus tard reste dans la période : deux lectures successives ne donnent le même ensemble que si `to` est assez ancienne pour qu'aucune exécution de la période ne soit encore en cours (chapitre 17, § 17.3.2), et sous réserve de la règle suivante
La sonde ne fournit **pas d'instantané** : les exécutions purgées par la rétention (chapitre 8, § 8.12) pendant la lecture disparaissent des pages suivantes	Pour un export complet, choisissez une borne `from` nettement plus récente que la limite de rétention, ou lisez les périodes anciennes en priorité

```bash
# Exécutions en échec d'un contrôle sur une période donnée
sapi "$SONDE/api/v1/runs?checkId=portail-accueil&status=CRITICAL&from=2025-11-18T00:00:00Z&to=2025-
    11-19T00:00:00Z&limit=50"
```

9.4 Utilisateurs et rôles

9.4.1 Les rôles

Les droits sont attribués par **rôle**. Chaque utilisateur et chaque jeton d'API possède un rôle unique.

Rôle	Droits	Usage recommandé
`admin`	Toutes les opérations, y compris comptes, jetons, secrets, réseau, mises à jour (politique, activation immédiate ou programmée, report), scripts et plugins personnalisés	Administrateurs de la sonde, en nombre limité

Rôle	Droits	Usage recommandé
`operator`	Créer et modifier les contrôles et plans **non scriptés**, activer, désactiver, déclencher, consulter. Peut **référencer** un secret existant dans un contrôle, sans pouvoir le lire, le créer, le modifier ni changer ses hôtes autorisés (`allowedHosts` , § 9.8). Pas de code exécutable : un `POST` , `PUT` ou `PATCH` d'un `operator` qui crée, modifie ou retire un scénario scripté (`browser.mode: script` ou `browser.scriptRef`) ou un plugin personnalisé est refusé avec `403` . Pas de paramètres système	Équipe d'exploitation, référents applicatifs
`viewer`	Lecture seule des états, résultats et artefacts, dont l'état de santé `updates` de la sonde (`GET /api/v1/system`)	Équipes support, consultation
`metrics`	Uniquement `/metrics` et l'état des contrôles (`GET /api/v1/checks/{id}/status`)	Prometheus, Nagios, Centreon

Information — Le téléversement de scripts de scénarios et de plugins personnalisés est réservé au rôle `admin` , parce qu'il revient à exécuter du code sur le réseau où la sonde est installée. Pour la même raison, seul un `admin` peut faire **désigner** un code par un contrôle : choisir ou changer le script d'un scénario (`scriptRef`) revient à décider du code exécuté. Un utilisateur `operator` peut créer des scénarios **déclaratifs** (chapitre 14), qui ne contiennent pas de code, et activer, désactiver ou déclencher un contrôle scripté existant.

Information — L'usage d'un secret est contrôlé **par la sonde**, et non par le seul rôle de l'auteur du contrôle. Chaque secret porte une liste obligatoire d'hôtes autorisés (`allowedHosts`), fixée par un `admin` (§ 9.8.1) : un `operator` qui référence un secret dans un contrôle ne peut le faire parvenir qu'à ces hôtes.

9.4.2 Gérer les comptes

Les comptes de l'interface d'administration sont des **comptes locaux** de la sonde. Le compte administrateur initial est créé lors de la mise en service (chapitre 6).

L'authentification repose uniquement sur ces comptes locaux. L'authentification par l'annuaire de l'entreprise (LDAP, SAML, OpenID Connect) n'est pas disponible.

Évolution prévue — L'authentification par **OpenID Connect** est prévue dans une version ultérieure. Le moteur d'authentification de la sonde repose sur des fournisseurs interchangeables (comptes locaux, puis OpenID Connect) : l'ajout de ce fournisseur, avec une correspondance entre les groupes de l'annuaire et les rôles de la sonde, ne modifiera ni les rôles décrits au § 9.4.1 ni l'API.

Objectif : créer un compte nominatif pour un membre de l'équipe d'exploitation.

Prérequis : rôle `admin` .

Étapes (interface d'administration) :

1. Ouvrez la gestion des comptes.
2. Créez un compte en indiquant l'identifiant, le rôle (`operator`) et un mot de passe initial conforme à la politique de mot de passe : 12 caractères au minimum et 128 au maximum, sans règle de composition imposée ni expiration périodique.
3. Transmettez le mot de passe initial par un canal sûr. Son changement est imposé à la première connexion, puisque le compte a été créé par un autre administrateur.

Étapes (API) : la création s'effectue par `POST /api/v1/users`, avec les champs `username`, `role` et `password`, décrits par le schéma `user.schema.json`. Dans l'exemple ci-dessous, le mot de passe est lu depuis l'entrée standard pour ne pas apparaître dans la ligne de commande.

```bash
read -rs MDP && printf '%s' "$MDP" | jq -Rs \
  '{username: "j.martin", role: "operator", password: .}' \
  | sapi -X POST -H 'Content-Type: application/json' --data @- \
      "$SONDE/api/v1/users"; unset MDP
```

Vérification : le compte apparaît dans `GET /api/v1/users`, et la création figure dans le journal d'audit.

Modifier le rôle ou le mot de passe d'un compte. Un administrateur modifie un compte existant par `PATCH /api/v1/users/{id}` (rôle `admin`), en indiquant le champ `role`, le champ `password`, ou les deux ; l'identifiant de connexion (`username`) ne se modifie pas. L'identifiant `{id}` du compte figure dans la réponse de `GET /api/v1/users`. Comme toute modification, la requête présente l'en-tête `If-Match` (§ 9.3.5), avec l'`ETag` obtenu par la lecture individuelle du compte, `GET /api/v1/users/{id}` (rôle `admin`). Cette lecture renvoie l'identifiant, l'identifiant de connexion et le rôle du compte, jamais son mot de passe ni l'empreinte de celui-ci. Exemple : réinitialiser le mot de passe d'un compte, lu sans affichage :

```bash
# 1. Lire l'ETag courant du compte
ETAG=$(sapi -D - -o /dev/null "$SONDE/api/v1/users/<id>" \
       | awk 'tolower($1)=="etag:" {print $2}' | tr -d '\r')

# 2. Définir le nouveau mot de passe
read -rs MDP && printf '%s' "$MDP" | jq -Rs '{password: .}' \
  | sapi -X PATCH -H 'Content-Type: application/merge-patch+json' \
      -H "If-Match: $ETAG" --data @- \
      "$SONDE/api/v1/users/<id>"; unset MDP
```

Dans cet exemple, `<id>` est l'identifiant du compte ; la première commande lit son `ETag` dans les en-têtes de la réponse de `GET /api/v1/users/{id}`. Le nouveau mot de passe respecte la même politique qu'à la création.

Effets d'une modification sur les sessions et les jetons du compte :

Modification	Sessions de l'interface du compte	Jetons d'API du compte
Changement de rôle (`role`)	Fermées immédiatement : l'utilisateur se reconnecte et obtient son nouveau rôle	**Inchangés** : chaque jeton conserve le rôle fixé à sa création, car le rôle est propre au jeton (§ 9.5.1)
Changement de mot de passe par un administrateur (`password`)	Fermées, à l'exception de la session depuis laquelle la requête est émise si elle appartient au compte modifié	Inchangés
Changement de son propre mot de passe (`PUT /api/v1/users/me/password`)	Les autres sessions sont fermées ; la session courante est conservée	Inchangés

> **Attention** — Abaisser le rôle d'un compte ne réduit pas les droits des jetons qu'il a créés. Si la modification doit s'appliquer aussi aux outils qui utilisent ces jetons, créez de nouveaux jetons du rôle voulu, puis révoquez les anciens (§ 9.5.4). De même, après un changement de mot de passe motivé par une compromission, révoquez les jetons du compte qui ont pu être exposés.

Changer son propre mot de passe. Tout utilisateur authentifié, quel que soit son rôle, change son propre mot de passe par `PUT /api/v1/users/me/password`, avec les champs `currentPassword` (mot de passe actuel) et `newPassword` (nouveau mot de passe, conforme à la politique ci-dessus). La saisie du mot de passe actuel empêche

qu'une session laissée ouverte suffise à changer le mot de passe. L'interface d'administration propose la même opération. Cette requête obéit aux règles suivantes :

- elle n'exige pas l'en-tête `If-Match` : la preuve demandée est le mot de passe actuel (exception au § 9.3.5) ;
- si le mot de passe actuel est erroné, la sonde répond `403 Forbidden` avec le type d'erreur `/api/v1/problems/current-password-invalid` (§ 9.3.4), et le mot de passe n'est pas modifié ;
- après le changement, les autres sessions de l'interface ouvertes pour ce compte sont fermées, la session courante est conservée, et les jetons d'API du compte ne sont pas modifiés.

```bash
read -rs ACTUEL && read -rs NOUVEAU \
  && printf '%s\n%s\n' "$ACTUEL" "$NOUVEAU" \
  | jq -Rn '[inputs] | {currentPassword: .[0], newPassword: .[1]}' \
  | sapi -X PUT -H 'Content-Type: application/json' --data @- \
      "$SONDE/api/v1/users/me/password"; unset ACTUEL NOUVEAU
```

Les deux mots de passe sont saisis l'un après l'autre, sans affichage, puis transmis à `jq` par l'entrée standard : comme dans les exemples précédents, ils n'apparaissent ni dans l'historique ni dans les arguments d'un processus.

Supprimer un compte. Utilisez `DELETE /api/v1/users/{id}`. Les sessions et jetons de l'utilisateur supprimé sont révoqués immédiatement.

9.4.3 Bonnes pratiques

- **Un compte par personne** : n'utilisez pas de compte partagé. Le journal d'audit n'a de valeur que si chaque action est rattachée à une personne.
- **Moindre privilège** : attribuez le rôle le plus restreint suffisant. La plupart des intervenants n'ont besoin que de `viewer` ou `operator`.
- **Au moins deux administrateurs** : pour éviter toute perte d'accès en cas d'absence ou d'oubli de mot de passe.
- **Code de récupération** : conservez dans le coffre-fort de l'organisation le code de récupération affiché une seule fois à la fin de la mise en service. Avec le code d'installation et un accès physique à la carte, il permet de redéfinir le mot de passe d'un compte `admin` sans perdre la configuration (chapitre 20). Sans lui, seule la réinitialisation usine permet de retrouver un accès administrateur.
- **Revue périodique** : supprimez les comptes des personnes ayant quitté l'équipe.

9.5 Jetons d'API

9.5.1 Principe

Un **jeton d'API** est une chaîne aléatoire qui authentifie un outil ou un script auprès de l'API. Il possède :

- un **rôle** qui limite ses droits (§ 9.4.1) ;
- un **nom** descriptif, qui indique son usage ;
- une **date d'expiration** facultative ;
- un **propriétaire**, l'utilisateur qui l'a créé.

La sonde ne conserve qu'une **empreinte** du jeton. La valeur en clair n'est affichée **qu'une seule fois**, au moment de la création : notez-la immédiatement dans un coffre-fort de mots de passe ou dans le gestionnaire de secrets de l'outil qui l'utilisera.

Un jeton est **immuable** : son nom, son rôle et sa date d'expiration ne se modifient plus après sa création, et l'API ne propose ni `PUT` ni `PATCH` sur les jetons. Pour changer l'un d'eux, ou pour renouveler un jeton qui arrive à expiration, créez un nouveau jeton, mettez à jour l'outil qui l'utilise, puis révoquez l'ancien (§ 9.5.4).

9.5.2 Créer un jeton

Objectif : créer un jeton en lecture seule pour le serveur Prometheus.

Prérequis : rôle `admin`.

Étapes :

1. Dans l'interface d'administration, ouvrez la gestion des jetons, ou utilisez `POST /api/v1/tokens`.
2. Indiquez un nom explicite (`prometheus-prod`), le rôle `metrics` et une date d'expiration conforme à votre politique de sécurité.
3. Copiez la valeur affichée et enregistrez-la dans l'outil destinataire.

Exemple par l'API (champs `name`, `role` et `expiresAt`) :

```bash
sapi -X POST -H 'Content-Type: application/json' \
  -d '{"name": "prometheus-prod", "role": "metrics", "expiresAt": "2026-11-30T00:00:00Z"}' \
  "$SONDE/api/v1/tokens"
```

Résultat attendu : la réponse contient la valeur du jeton, visible cette seule fois. Les champs de la requête et de la réponse sont décrits par le schéma `token.schema.json`.

Vérification : le jeton apparaît dans `GET /api/v1/tokens`, sans sa valeur.

9.5.3 Choisir le rôle et la durée de vie

Usage	Rôle	Expiration conseillée
Prometheus (collecte `/metrics`)	`metrics`	12 mois, renouvellement planifié
Nagios, Centreon (`check_sva`)	`metrics`	12 mois
Tableau de bord interne en lecture	`viewer`	12 mois
Outil d'automatisation créant des contrôles	`operator`	3 à 6 mois
Outil d'automatisation complet (déploiement de scripts, secrets)	`admin`	Courte ; à réserver à des outils de confiance

9.5.4 Révoquer un jeton

Révoquez immédiatement un jeton par `DELETE /api/v1/tokens/{id}` ou depuis l'interface lorsque :

- l'outil qui l'utilisait est retiré ;
- le jeton a pu être exposé (fichier partagé par erreur, dépôt de code, capture d'écran) ;
- son propriétaire quitte l'équipe.

La révocation est effective immédiatement : les appels suivants reçoivent `401 Unauthorized`.

> ! **Avertissement** — Ne stockez jamais un jeton d'API dans un dépôt de code source, un fichier de configuration lisible par tous ou un ticket. Utilisez le gestionnaire de secrets de votre outil d'automatisation (coffre Ansible, gestionnaire de secrets de votre chaîne d'intégration continue, coffre-fort d'entreprise).

9.6 Gestion des contrôles

Cette section décrit les opérations sur les contrôles. Les options propres à chaque type sont détaillées :

- au chapitre 10 pour `icmp`, `tcp`, `dns`, `http` et `nagios` ;
- au chapitre 11 pour `api` et `database` ;
- au chapitre 14 pour `browser`.

9.6.1 Opérations disponibles

Opération	Méthode et chemin	Rôle minimal
Lister les contrôles	`GET /api/v1/checks` (filtres `type`, `enabled`, `tag`, `status`)	`viewer`
Lire un contrôle	`GET /api/v1/checks/{id}`	`viewer`
Créer	`POST /api/v1/checks`	`operator` ; `admin` pour un scénario scripté ou un contrôle qui utilise un plugin personnalisé
Remplacer	`PUT /api/v1/checks/{id}`	`operator` pour un contrôle non scripté ; `admin` si le contrôle existant ou la nouvelle définition est un scénario scripté ou utilise un plugin personnalisé, même si son code reste inchangé (un `operator` ajuste la planification ou les seuils d'un tel contrôle par `PATCH`)
Modifier partiellement	`PATCH /api/v1/checks/{id}` (JSON Merge Patch)	`operator` ; `admin` si la modification crée, change ou retire `browser.mode: script`, `browser.scriptRef` ou un plugin personnalisé
Supprimer	`DELETE /api/v1/checks/{id}`	`operator`
Activer, désactiver	`POST /api/v1/checks/{id}/enable`, `POST /api/v1/checks/{id}/disable`	`operator`
Exécuter immédiatement	`POST /api/v1/checks/{id}/runs`	`operator`
Exécution d'essai	`POST /api/v1/checks/{id}/runs?dryRun=true`	`operator`
État courant	`GET /api/v1/checks/{id}/status`	`metrics`

> **Attention** — Un `operator` peut ajuster la planification, les seuils ou les étiquettes d'un scénario scripté par `PATCH`, mais toute requête de sa part qui touche au code exécuté (passage en mode `script`, changement ou retrait de `scriptRef`, ajout, changement ou retrait d'un plugin personnalisé) est refusée avec `403`. Le choix du code exécuté relève toujours d'un `admin` (§ 9.4.1).

9.6.2 Créer un contrôle

Objectif : créer un contrôle HTTP vérifiant la page d'accueil d'un portail.

Prérequis : rôle `operator` ; édition Essential ou supérieure ; la cible est joignable depuis la sonde (chapitre 8, § 8.2.2).

Étapes :

1. Rédigez la définition du contrôle dans un fichier `portail-accueil.json` :

```json
{
  "id": "portail-accueil",
  "name": "Portail client : page d'accueil",
  "type": "http",
  "enabled": false,
  "tags": { "application": "portail", "environment": "production" },
  "schedule": { "interval": "1m", "jitter": "10s" },
  "timeout": "15s",
  "retry": { "attempts": 1, "interval": "20s" },
  "thresholds": { "warningDuration": "3s" },
  "http": {
    "method": "GET",
    "url": "https://portail.example.com/",
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "body", "operator": "contains", "value": "Espace client" }
    ]
  }
}
```

   Le contrôle est créé **désactivé** (`"enabled": false`) pour pouvoir être testé avant d'entrer en service.

2. Envoyez-le à la sonde :

```bash
sapi -X POST -H 'Content-Type: application/json' \
  --data @portail-accueil.json "$SONDE/api/v1/checks"
```

3. Lancez une exécution d'essai (§ 9.6.5) et vérifiez son résultat.
4. Activez le contrôle (§ 9.6.4).

Résultat attendu : réponse `201 Created` ; le contrôle apparaît dans la liste et dans l'interface.

En cas de problème : une réponse `422` détaille les champs non conformes (§ 9.3.4) ; une réponse `409` indique que l'identifiant existe déjà ; une réponse `403` signale un rôle insuffisant ou, avec le type d'erreur `/api/v1/problems/edition-feature-unavailable`, un type de contrôle non inclus dans votre édition (§ 9.3.4).

Dans l'interface d'administration, la même opération s'effectue par un formulaire. Celui-ci applique les mêmes règles de validation que l'API (§ 9.3.4).

9.6.3 Modifier un contrôle

Deux méthodes sont possibles :

- `PUT` remplace la définition complète. C'est la méthode à privilégier lorsque la définition de référence est conservée dans un fichier, par exemple dans un dépôt de configuration ;
- `PATCH` ne modifie que les champs fournis (format JSON Merge Patch, RFC 7396). C'est la méthode la plus pratique pour un ajustement ponctuel.

Exemple : porter la fréquence à 2 minutes, en respectant le contrôle de concurrence (§ 9.3.5).

```bash
# 1. Lire l'ETag courant
ETAG=$(sapi -D - -o /dev/null "$SONDE/api/v1/checks/portail-accueil" \
       | awk 'tolower($1)=="etag:" {print $2}' | tr -d '\r')

# 2. Appliquer la modification partielle
sapi -X PATCH -H 'Content-Type: application/merge-patch+json' \
  -H "If-Match: $ETAG" \
  -d '{"schedule": {"interval": "2m", "jitter": "15s"}}' \
  "$SONDE/api/v1/checks/portail-accueil"
```

Chaque modification incrémente la **révision** du contrôle. Le numéro de révision est enregistré dans chaque résultat d'exécution : vous savez ainsi quelle version de la configuration a produit un résultat donné.

> **Information** — Avec JSON Merge Patch, les objets sont fusionnés champ par champ, mais un **tableau** fourni remplace intégralement le tableau existant. Pour modifier une seule assertion d'un contrôle HTTP, renvoyez donc la liste complète des assertions. Une propriété fixée à `null` est **supprimée** : `{"schedule": null}` retire ainsi la planification propre d'un contrôle, qui ne s'exécute plus que par un plan d'exécution ou manuellement (chapitre 12).

9.6.4 Activer et désactiver

Désactiver un contrôle suspend ses exécutions planifiées sans le supprimer, et conserve son historique. C'est la méthode à utiliser pendant une intervention planifiée sur l'application supervisée, pour éviter des alertes inutiles.

```bash
sapi -X POST "$SONDE/api/v1/checks/portail-accueil/disable"
sapi -X POST "$SONDE/api/v1/checks/portail-accueil/enable"
```

> **Conseil** — Notez dans votre procédure de changement de **réactiver** les contrôles à la fin de l'intervention. Un contrôle oublié en état désactivé ne supervise plus rien. La liste `GET /api/v1/checks?enabled=false` permet de les repérer.

> **Évolution prévue** — Des fenêtres de maintenance planifiées, suspendant automatiquement des contrôles sur une période donnée, sont prévues dans une version ultérieure (chapitre 12, § 12.8).

9.6.5 Exécution immédiate et exécution d'essai

	Exécution immédiate	Exécution d'essai (`dryRun`)
Appel	`POST /api/v1/checks/{id}/runs`	`POST /api/v1/checks/{id}/runs?dryRun=true`
Contrôle désactivé	Autorisé, par exemple pour mettre au point un contrôle avant de l'activer	Autorisé
Effet sur l'état courant	Oui	Non
Export vers les plateformes	Oui	Non
Conservation du résultat	Dans l'historique	Consultable, marqué `dryRun`

	Exécution immédiate	Exécution d'essai (`dryRun`)
Usage	Confirmer un incident ou la résolution d'un incident, rejouer un contrôle	Mettre au point un nouveau contrôle ou une modification sans perturber la supervision

L'exécution étant asynchrone, la sonde répond `202 Accepted` avec un en-tête `Location` vers la ressource d'exécution. Il suffit de suivre cette ressource jusqu'à la fin de l'exécution :

```bash
# Lancer une exécution d'essai et récupérer l'adresse de suivi
LOC=$(sapi -X POST -D - -o /dev/null \
      "$SONDE/api/v1/checks/portail-accueil/runs?dryRun=true" \
      | awk 'tolower($1)=="location:" {print $2}' | tr -d '\r')

# Consulter le résultat (répéter jusqu'à la fin de l'exécution)
sapi "$SONDE$LOC" | jq '{status, durationMs, error}'
```

L'en-tête `Location` de la réponse `202 Accepted` désigne la ressource de l'exécution, de la forme `/api/v1/runs/{runId}`.

> ✓ **Conseil** — Réalisez systématiquement une exécution d'essai après toute création ou modification significative d'un contrôle. Elle valide en une opération la configuration, l'accès réseau à la cible et les assertions.

9.6.6 Supprimer un contrôle

```bash
ETAG=$(sapi -D - -o /dev/null "$SONDE/api/v1/checks/portail-accueil" \
       | awk 'tolower($1)=="etag:" {print $2}' | tr -d '\r')
sapi -X DELETE -H "If-Match: $ETAG" "$SONDE/api/v1/checks/portail-accueil"
```

La suppression est définitive pour la configuration. L'historique des exécutions reste consultable jusqu'à l'échéance de sa durée de rétention. La suppression d'un contrôle référencé par un plan d'exécution est refusée (`409 Conflict`) ; la réponse liste les plans concernés. Retirez d'abord le contrôle de ces plans.

> ! **Attention** — Recréer ultérieurement un contrôle avec le même identifiant le rattache, dans les plateformes d'observabilité, à l'historique de l'ancien contrôle. Si la nouvelle définition vérifie autre chose, choisissez un nouvel identifiant.

9.7 Scripts de scénarios

Les scénarios **scriptés** (édition Enterprise) sont des modules JavaScript Puppeteer, décrits au chapitre 14. Le script est téléversé séparément du contrôle, puis référencé par celui-ci.

9.7.1 Téléverser un script

Le téléversement d'un script se fait par l'interface d'administration ou par `POST /api/v1/scripts` (rôle `admin`, édition Enterprise). La procédure complète de mise en service d'un scénario scripté, du téléversement à l'activation du contrôle, figure au chapitre 14 (§ 14.16.2). Les règles propres à l'administration sont les suivantes :

- chaque requête transmet un seul module `.mjs`, avec le type de contenu `text/javascript`, dans la limite de 1 Mo. Les scénarios déclaratifs ne passent pas par cette ressource : leurs étapes figurent dans la définition du contrôle (`POST /api/v1/checks`, type `application/json`) ;

- la sonde valide le script, calcule son empreinte SHA-256 et renvoie une **référence** de la forme `nom@sha256:empreinte`, par exemple `boutique-connexion@sha256:4f1c9e2ab7d3` (empreinte abrégée dans les exemples de cette documentation : une référence réelle comporte les 64 caractères hexadécimaux de l'empreinte), à indiquer dans la propriété `browser.scriptRef` du contrôle ;
- le téléversement d'un script et toute création, modification ou suppression de `browser.scriptRef` dans un contrôle sont réservés au rôle `admin` : la même requête émise par un `operator` est refusée avec `403` (§ 9.4.1 et § 9.6.1).

Vérification : `GET /api/v1/scripts/{name}` liste les versions téléversées.

9.7.2 Pourquoi une référence par empreinte

La référence désigne **une version précise et immuable** du script. Téléverser une nouvelle version ne modifie donc pas les contrôles existants : le changement n'intervient qu'au moment où vous mettez à jour `scriptRef` dans le contrôle. Ce mécanisme apporte trois garanties :

- aucune modification involontaire d'un scénario en production ;
- la possibilité de revenir à la version précédente en rétablissant l'ancienne référence ;
- la traçabilité : le journal d'audit indique qui a téléversé quelle version, et quand.

> ✓ **Conseil** — Conservez vos scripts dans un système de gestion de versions (Git), et faites figurer l'empreinte renvoyée par la sonde dans le message de version. Le lien entre le code source et ce qui s'exécute sur la sonde reste ainsi vérifiable.

9.8 Secrets

9.8.1 Principe

Un **secret** est une valeur sensible dont un contrôle a besoin : mot de passe d'un compte de test, jeton d'une API, mot de passe d'une base de données. La sonde stocke les secrets **chiffrés** et applique quatre règles :

1. **Écriture seule** : un secret peut être créé, remplacé ou supprimé, mais sa valeur ne peut jamais être relue, ni par l'interface ni par l'API, quel que soit le rôle.
2. **Déclaration explicite** : un contrôle ne reçoit que les secrets qu'il déclare dans sa propriété `secrets`.
3. **Masquage** : la valeur d'un secret est masquée dans les messages d'erreur, les journaux, les sorties de plugins et la console du navigateur enregistrée.
4. **Destinations autorisées** : chaque secret porte une liste **obligatoire** d'hôtes autorisés, `allowedHosts` (noms d'hôtes ou motifs de noms, avec un port éventuel), fixée par un `admin`. La sonde n'injecte un secret que dans une requête, une connexion ou une page dont l'hôte figure dans cette liste.

Cette quatrième règle protège contre le détournement d'un secret : un contrôle non scripté modifié pour envoyer le secret vers un autre serveur ne le reçoit pas ; le cas des scénarios scriptés est précisé ci-dessous. Elle s'applique ainsi :

Situation	Comportement de la sonde
L'hôte de la requête, de la connexion ou de la page figure dans `allowedHosts`	Le secret est injecté
L'hôte n'y figure pas	Le secret n'est pas injecté ; l'exécution échoue

Situation	Comportement de la sonde
Redirection HTTP vers un **autre hôte**	Les en-têtes porteurs d'un secret (par exemple `Authorization`) ne sont **pas retransmis** à la destination de la redirection. Une redirection étant une nouvelle requête, la règle d'injection s'applique à sa destination pour **tout élément porteur d'un secret** (en-tête, corps renvoyé lors d'une redirection `307` ou `308`, paramètre d'adresse), et aucun n'est transmis à un hôte absent de la liste
Hôte inscrit avec un port (par exemple `db.example.internal:5432`)	Le secret n'est injecté que vers ce port
Changement de destination (URL ou hôte modifié dans le contrôle, navigation vers une autre page)	La vérification porte sur la destination de chaque requête, connexion ou page au moment de l'injection : un contrôle modifié pour viser un hôte absent de la liste ne reçoit pas le secret
Scénario scripté (`browser.mode=script`)	Le script reçoit la valeur dans l'objet `secrets` (chapitre 14) : la sonde ne peut pas garantir à elle seule que le code ne la transmet pas ailleurs. C'est l'une des raisons pour lesquelles ces scénarios sont réservés au rôle `admin`
Un `operator` crée ou modifie un contrôle	Il peut référencer un secret existant ; il ne peut ni lire sa valeur ni modifier `allowedHosts`
Import d'une configuration (§ 9.12.3)	Les références aux secrets sont **revérifiées** : secret existant, déclaration dans le contrôle, hôtes autorisés
Import d'une archive antérieure à `allowedHosts` (secret sans liste)	Le secret est importé mais reste **inutilisable**, c'est-à-dire jamais injecté, tant qu'un `admin` n'a pas défini sa liste ; le compte rendu d'import le signale

Les scénarios scriptés restent réservés au rôle `admin` (§ 9.4.1).

Les noms de secrets s'écrivent en majuscules, chiffres et tirets bas, et commencent par une lettre (`BOUTIQUE_PASSWORD`, `API_STOCK_TOKEN`).

9.8.2 Créer ou remplacer un secret

Objectif : enregistrer le mot de passe du compte de test de la boutique.

Prérequis : rôle `admin`.

Étapes :

1. Dans l'interface d'administration, ouvrez la gestion des secrets et créez le secret `BOUTIQUE_PASSWORD`, ou utilisez l'API, `PUT /api/v1/secrets/{name}`, avec un corps de la forme `{"value": …, "allowedHosts": [ … ]}`. La liste `allowedHosts` est obligatoire et contient au moins un hôte : ici, le seul site de la boutique de test (nom fictif). Pour une **création**, la requête porte l'en-tête `If-None-Match: *` (§ 9.3.5) : si un secret de même nom existe déjà, la sonde répond `412 Precondition Failed` au lieu de l'écraser. Exemple, la valeur étant lue sans affichage :

   ```bash
   read -rs VAL && printf '%s' "$VAL" \
     | jq -Rs '{value: ., allowedHosts: ["recette.boutique.example.com"]}' \
     | sapi -X PUT -H 'Content-Type: application/json' -H 'If-None-Match: *' \
         --data @- "$SONDE/api/v1/secrets/BOUTIQUE_PASSWORD"; unset VAL
   ```

2. Vérifiez sa présence : `GET /api/v1/secrets` (rôle `admin`) liste les noms et les métadonnées des secrets, jamais leurs valeurs. Ces métadonnées comprennent la date de rotation et la liste `allowedHosts` de chaque secret. Pour un seul secret, `GET /api/v1/secrets/{name}` (rôle `admin`) renvoie ses métadonnées, dont `allowedHosts`, et son `ETag`, nécessaires à sa rotation (§ 9.8.4) ; cette lecture ne renvoie jamais la valeur du secret.

Corps de requête correspondant (valeur fictive) :

```json
{ "value": "********", "allowedHosts": ["recette.boutique.example.com"] }
```

Résultat attendu : le secret est disponible pour les contrôles qui le déclarent, et uniquement vers les hôtes autorisés.

9.8.3 Utiliser un secret dans un contrôle

Un secret s'utilise en deux temps :

1. il est **déclaré** dans la liste `secrets` du contrôle ;
2. il est **référencé** à l'endroit voulu par la syntaxe `${secret:NOM}`. Dans un scénario scripté, il est lu dans l'objet `secrets` (chapitre 14).

```json
{
  "id": "api-stock",
  "name": "API de stock : article témoin",
  "type": "http",
  "schedule": { "interval": "2m" },
  "timeout": "10s",
  "secrets": ["API_STOCK_TOKEN"],
  "http": {
    "method": "GET",
    "url": "https://api.example.com/v1/stock/ARTICLE-TEMOIN",
    "headers": { "Authorization": "Bearer ${secret:API_STOCK_TOKEN}" },
    "assertions": [ { "kind": "status", "operator": "eq", "value": 200 } ]
  }
}
```

Dans cet exemple, le secret `API_STOCK_TOKEN` doit avoir été créé avec `"allowedHosts": ["api.example.com"]` : l'hôte de l'URL du contrôle figure dans la liste, le secret est donc injecté dans l'en-tête `Authorization`.

Un contrôle qui référence un secret non déclaré, ou un secret inexistant, est refusé à la création (`422`). La suppression d'un secret encore référencé par un contrôle est refusée (`409`).

9.8.4 Rotation

Remplacer la valeur d'un secret suffit : tous les contrôles qui l'utilisent emploient la nouvelle valeur dès leur prochaine exécution. Le remplacement utilise la même requête qu'à la création (§ 9.8.2), mais avec l'en-tête `If-Match` portant l'`ETag` du secret, et non `If-None-Match: *` (§ 9.3.5). Chaque `PUT` exige en outre la liste `allowedHosts` en même temps que la valeur. Lisez donc d'abord le secret par `GET /api/v1/secrets/{name}` (rôle `admin`) : la réponse contient ses métadonnées, dont la liste `allowedHosts`, et l'en-tête `ETag`, mais jamais sa valeur. Renvoyez la liste à l'identique, sauf modification voulue, pour ne pas restreindre ni étendre par mégarde les destinations autorisées. La date de rotation est enregistrée et visible dans la liste des secrets.

Exemple de rotation, la nouvelle valeur étant lue sans affichage (nom de secret fictif) :

```bash
# 1. Lire les métadonnées et l'ETag du secret (la valeur n'est jamais renvoyée)
ENTETES=$(mktemp)
META=$(sapi -D "$ENTETES" "$SONDE/api/v1/secrets/BOUTIQUE_PASSWORD")
ETAG=$(awk 'tolower($1)=="etag:" {print $2}' "$ENTETES" | tr -d '\r'); rm -f "$ENTETES"

# 2. Remplacer la valeur en renvoyant à l'identique la liste allowedHosts relue
read -rs VAL && printf '%s' "$VAL" \
  | jq -Rs --argjson hotes "$(jq -c '.allowedHosts' <<<"$META")" \
      '{value: ., allowedHosts: $hotes}' \
  | sapi -X PUT -H 'Content-Type: application/json' -H "If-Match: $ETAG" \
      --data @- "$SONDE/api/v1/secrets/BOUTIQUE_PASSWORD"; unset VAL
```

Si un autre administrateur a modifié le secret entre la lecture et le remplacement, la sonde répond `412 Precondition Failed` : relisez le secret et recommencez.

Procédure de rotation sans interruption d'un mot de passe de compte de test :

1. Prévenez les équipes concernées et choisissez un moment hors période critique.
2. Modifiez le mot de passe du compte de test dans l'application supervisée.
3. Remplacez immédiatement la valeur du secret sur la sonde (exemple ci-dessus).
4. Lancez une exécution immédiate des contrôles concernés (§ 9.6.5) pour confirmer le bon fonctionnement.

Entre les étapes 2 et 3, une exécution peut échouer avec le type d'erreur `auth_failed`. Si l'application le permet, désactivez temporairement les contrôles concernés pendant l'opération.

9.8.5 Supprimer un secret

`DELETE /api/v1/secrets/{name}` supprime un secret, avec l'en-tête `If-Match` portant l'`ETag` renvoyé par `GET /api/v1/secrets/{name}` (§ 9.3.5). La suppression est refusée (`409 Conflict`) tant qu'un contrôle le référence.

> **Avertissement** — Utilisez des **comptes de test dédiés** à la supervision, dotés des droits strictement nécessaires, et jamais des comptes nominatifs ou des comptes à privilèges. Pour les bases de données, privilégiez un compte en **lecture seule**. Une sonde est un équipement installé dans votre réseau : limiter ce que ses identifiants permettent limite les conséquences d'une compromission.

9.9 Paramètres de la sonde

Les paramètres système sont regroupés en sections, consultables et modifiables par un administrateur dans l'interface ou par `GET` et `PUT /api/v1/settings/{section}`.

Section	Contenu	Détail
`network`	Nom d'hôte, adressage DHCP ou statique, passerelle, serveurs DNS (IPv4 ou IPv6), publication mDNS, domaines de recherche DNS (`searchDomains`) ; objet `ipv6` : `enabled` (`false` par défaut), `mode` (`slaac` ou `static`), `address` (notation CIDR), `gateway`	Chapitre 7
`dot1x`	Authentification 802.1X de la prise Ethernet par certificat (EAP-TLS) : `enabled`, `identity`, `caRef` (autorité de certification du serveur RADIUS, importée dans la section `trust`). L'identifiant client (clé et certificat) se gère par les opérations du § 9.9.3	Chapitre 7
`time`	Serveurs NTP, fuseau horaire d'affichage	Chapitre 7
`proxy`	Proxy HTTP(S) sortant, authentification, exclusions	Chapitre 7

Section	Contenu	Détail
`retention`	Durées de conservation des exécutions, agrégats, artefacts, audit	Ci-dessous
`security`	Politique de session, verrouillage après échecs, liste des réseaux autorisés à administrer (`allowedNetworks` , IPv4 ou IPv6), redirection du port 80 (`httpRedirect`)	Chapitre 7
`ssh`	Activation temporaire du service SSH : `enabled` , `enabledUntil` (obligatoire à l'activation ; 4 h par défaut, 72 h au maximum), `authorizedKeys` (clés publiques autorisées), `locked` (verrouillage définitif : voir § 9.9.3)	Chapitre 7
`updates`	Politique de mises à jour : `security.mode` (`automatic` par défaut, ou `manual`) et `security.criticalPolicy` (`next-window` par défaut, ou `asap`) pour les correctifs de sécurité, avec `security.criticalStagger` (décalage fixe des activations `asap` , `0` par défaut, `4h` au maximum, chapitre 20, § 20.4.11) ; `feature.mode` (`manual` par défaut, ou `automatic`) pour les versions fonctionnelles ; `windows` : 0 à 7 fenêtres de maintenance (`days` , `start` au format `HH:MM` , `duration`), **aucune par défaut** ; `notice` : préavis avant activation (`24h` par défaut, de `0` à `7d`) ; `drainTimeout` : délai de drainage (`5m` par défaut, `30m` au maximum) ; `repositoryUrl` : miroir interne facultatif	Chapitre 20, § 20.4.2 et § 20.4.8
`license`	Lecture seule : édition, numéro de série, état du certificat d'édition ; aucune clé à saisir	Chapitre 6, § 6.5
`trust`	Autorités de certification internes reconnues par la sonde (format PEM)	Chapitre 7, § 7.7.3
`logging`	Persistance temporaire des journaux système (`persistentUntil` , 72 h au plus)	Chapitre 19

9.9.1 Modifier la configuration réseau : confirmation obligatoire

Une erreur de configuration réseau peut rendre la sonde injoignable. Pour éviter cette situation, une nouvelle configuration réseau s'applique en deux temps :

1. Enregistrez les paramètres (`PUT /api/v1/settings/network`), puis demandez leur application (`POST /api/v1/settings/network/apply`), qui renvoie un identifiant.
2. Reconnectez-vous à la sonde, à sa nouvelle adresse le cas échéant, et **confirmez** la configuration dans un délai de **120 secondes**.

Sans confirmation dans ce délai, la sonde revient automatiquement à la configuration précédente. Par l'API, la confirmation s'effectue par `POST /api/v1/settings/network/confirm` , appelé **depuis la nouvelle adresse** ; l'interface d'administration l'envoie automatiquement. La procédure complète figure au chapitre 7.

9.9.2 Rétention des données

Les durées de conservation par défaut sont présentées au chapitre 8 (§ 8.12.1). Elles se règlent dans la section `retention` . Exemple :

```json
{
  "runs": "30d",
  "hourlyAggregates": "400d",
  "artifacts": { "maxAge": "7d", "maxSize": "500MB" },
  "audit": "365d"
}
```

> **Attention** — Allonger la rétention des exécutions détaillées augmente l'espace occupé et le volume d'écritures sur la carte microSD. Avant de dépasser 30 jours, vérifiez l'espace disponible dans l'état de santé de la sonde (chapitre 20). Pour une analyse de long terme, préférez les agrégats horaires ou une plateforme d'observabilité (chapitre 18).

9.9.3 Opérations système

Certaines opérations d'administration ne modifient pas une section de paramètres : elles déclenchent une action ou importent un élément signé. Elles sont toutes réservées au rôle `admin` et enregistrées dans le journal d'audit (§ 9.10). Le tableau suivant les recense ; leurs procédures complètes figurent dans les chapitres indiqués.

Opération	Appel	Détail
Consulter l'état des mises à jour : version de chaque emplacement système, emplacement actif, mises à jour disponibles, fin de la période de support (`supportEndsAt`) et du justificatif de maintenance (`maintenanceEndsAt`) ; état de la mise à jour en attente (`state`), activation annoncée (`scheduledActivation`) et report en cours (`postponedUntil`)	`GET /api/v1/updates`	Chapitre 20
Activer la mise à jour installée sur l'emplacement inactif, immédiatement, ou à une heure choisie avec le corps facultatif `{"scheduledAt": "<date RFC 3339>"}`	`POST /api/v1/updates/apply`	Chapitre 20, § 20.4.3
Reporter l'activation prévue jusqu'à une date donnée : corps `{"until": "<date RFC 3339>"}` ; 30 jours au plus pour un correctif ordinaire, 7 jours au plus pour un correctif critique	`POST /api/v1/updates/postpone`	Chapitre 20, § 20.4.2
Importer le justificatif de maintenance, fichier signé distinct du certificat d'édition	`PUT /api/v1/system/maintenance`	Chapitre 20
Activer le SSD NVMe facultatif (chapitre 20, § 20.3.4) . Le corps de la requête confirme l'opération par le nom d'hôte de la sonde : `{"confirm": "<nom d'hôte>"}`	`POST /api/v1/system/storage/ssd`	Chapitre 20
Générer sur la sonde la clé et la demande de certificat (CSR) de l'identifiant client 802.1X (méthode recommandée)	`POST /api/v1/system/dot1x/csr`	Chapitre 7
Installer le certificat client signé (PEM) émis pour la dernière CSR, ou importer un PKCS#12 protégé (écriture seule : l'identifiant ne peut pas être relu)	`PUT /api/v1/system/dot1x/credential`	Chapitre 7
Verrouiller définitivement SSH, avec confirmation par le nom d'hôte : `{"confirm": "<nom d'hôte>"}`	`POST /api/v1/settings/ssh/lock`	Chapitre 7

> **Attention** — L'activation du SSD **formate** le SSD raccordé, migre les données de la sonde, puis la redémarre. Prévoyez une interruption des contrôles pendant la migration et le redémarrage. Le démarrage reste sur la microSD.

> **Avertissement** — Le verrouillage de SSH est **irréversible** : il ne peut être levé que par une réinitialisation usine, qui efface la configuration. Une fois SSH verrouillé, le support ne pourra plus intervenir par SSH ; la récupération du compte administrateur par code de récupération (§ 9.4.3) reste possible. Le verrouillage peut aussi être demandé dès la mise en service, par le fichier de provisionnement (`ssh.locked: true` , chapitre 6).

> **Information** — L'activation d'une mise à jour, c'est-à-dire le redémarrage sur la nouvelle version, n'a lieu que dans une fenêtre de maintenance définie dans la section `updates`, ou à la demande d'un administrateur (`POST /api/v1/updates/apply`, immédiatement ou à l'heure `scheduledAt`). Elle commence par un **état de maintenance** : aucune exécution nouvelle ne démarre, celles en cours disposent du délai `drainTimeout`, puis sont interrompues en `UNKNOWN` avec le type d'erreur `maintenance`, et `GET /health` répond `{"status": "MAINTENANCE"}` (code `503`). Cet état ne suffit pas à neutraliser les alertes de vos plateformes, car la sonde est injoignable pendant son redémarrage. Procédures complètes : chapitre 20 (§ 20.4.3, § 20.4.8 à § 20.4.10) ; neutralisation des alertes : chapitre 18.

9.10 Journal d'audit

Le **journal d'audit** enregistre toutes les opérations d'administration, avec pour chacune :

- la date ;
- l'auteur, utilisateur ou jeton ;
- l'adresse d'origine ;
- l'action, l'objet concerné et le résultat ;
- pour une modification, la différence entre l'ancienne et la nouvelle configuration, **sans les valeurs des secrets**.

Événements enregistrés :

Catégorie	Exemples
Authentification	Connexions réussies et échouées, blocages
Configuration	Création, modification, suppression de contrôles, plans, destinations d'export, paramètres
Secrets	Création, rotation, suppression (sans valeur)
Code exécutable	Téléversement de scripts et de plugins
Comptes et jetons	Création, suppression, révocation
Système	Mises à jour, réinitialisation, activation, désactivation et verrouillage de SSH, connexions SSH, import de configuration, import du justificatif de maintenance, activation du SSD, gestion de l'identifiant 802.1X, récupération d'un compte administrateur
Mises à jour et maintenance	Modification de la section `updates` ; annonce d'une activation (événement `update.scheduled`), qu'elle provienne d'une fenêtre ou d'une demande d'un administrateur (`scheduledAt`) ; report d'une activation (`POST /api/v1/updates/postpone`) ; début et fin de l'état de maintenance ; résultat de l'activation, retour arrière compris

Le journal se consulte dans l'interface ou par `GET /api/v1/audit-events` (rôle `admin`). Il est conservé 365 jours par défaut. Il peut aussi être transmis sous forme de logs à votre plateforme d'observabilité ou à votre outil de gestion des événements de sécurité (SIEM), par une destination d'export OpenTelemetry dont l'option `logs.audit` est fixée à `true` (chapitre 18). Cet export est désactivé par défaut.

```bash
# Opérations de la dernière journée (filtres disponibles : from, to, actor, action)
sapi "$SONDE/api/v1/audit-events?from=2025-11-18T00:00:00Z&limit=100" | jq '.items[]'
```

9.11 Événements en temps réel

L'adresse `GET /api/v1/events` diffuse les événements de la sonde en temps réel, selon le protocole **Server-Sent Events** (SSE). Une connexion HTTP reste ouverte, et la sonde y envoie chaque événement dès qu'il se produit.

Événement	Émis quand
`run.started`	Une exécution démarre
`run.finished`	Une exécution se termine, avec son état
`check.status.changed`	L'état courant d'un contrôle change, par exemple de `OK` à `CRITICAL`
`system.alert`	La sonde signale une anomalie la concernant : espace disque, mémoire, défaillance d'un composant, échec d'export

Exemple d'écoute depuis un terminal (l'option `-N` désactive la mise en mémoire tampon) :

```bash
sapi -N -H 'Accept: text/event-stream' "$SONDE/api/v1/events"
```

L'interface d'administration utilise ce flux pour actualiser le tableau de bord sans rechargement. Il peut aussi alimenter un écran de supervision local ou un outil interne.

> **Information** — Ce flux est destiné à l'observation en direct. Il n'est pas conçu pour l'archivage : un événement émis pendant une déconnexion n'est pas renvoyé. Pour l'historique, utilisez `GET /api/v1/runs` ou une plateforme d'observabilité.

9.12 Sauvegarde et restauration de la configuration

9.12.1 Ce qui est sauvegardé

La sonde réalise **automatiquement** une sauvegarde quotidienne de sa configuration et en conserve trois générations. Ces sauvegardes, stockées sur la sonde, lui permettent de se rétablir seule en cas de corruption de sa base (chapitre 20). Elles ne protègent pas contre la perte ou la panne de la carte microSD : réalisez donc aussi des **exports** conservés hors de la sonde.

Contenu de l'export	Inclus
Contrôles, plans d'exécution, scripts	Oui
Destinations d'export	Oui
Paramètres (heure, proxy, rétention, sécurité, autorités de certification `trust`, mises à jour, journaux, SSH)	Oui, **sauf** la section `network`, propre à chaque appareil ; le verrouillage SSH (`ssh.locked`) est ignoré à l'import
Comptes utilisateurs	Inclus, sous forme d'empreintes, uniquement avec une phrase de passe
Secrets	**Exclus**, ou inclus **chiffrés** (AES-256-GCM, clé dérivée par Argon2id) avec une phrase de passe que vous fournissez. Chaque secret y est conservé avec sa liste `allowedHosts`, restaurée à l'import
Historique des exécutions et artefacts	Non

9.12.2 Exporter

Objectif : conserver une copie de la configuration hors de la sonde.

Prérequis : rôle `admin`.

Étapes :

1. Lancez l'export depuis l'interface, ou par l'API :

   ```bash
   sapi -o "sonde-paris-01-config-$(date +%Y%m%d).json" \
     "$SONDE/api/v1/config/export"
   ```

   Pour inclure les comptes et les secrets, chiffrés, envoyez la demande d'export en `POST` avec la phrase de passe dans l'en-tête `X-Config-Passphrase`, comme pour l'import (§ 9.12.3), et jamais dans l'adresse, qui peut être journalisée. La phrase de passe est lue sans affichage, puis placée dans un fichier d'en-tête temporaire, lisible par vous seul et supprimé aussitôt, pour ne figurer dans les arguments d'aucune commande (même méthode qu'au § 9.3.3) :

   ```bash
   (
     umask 077
     ENTETE=$(mktemp) && trap 'rm -f "$ENTETE"' EXIT &&
     IFS= read -rs PHRASE &&
     printf 'X-Config-Passphrase: %s\n' "$PHRASE" > "$ENTETE" &&
     unset PHRASE &&
     sapi -X POST -H @"$ENTETE" \
       -o "sonde-paris-01-config-$(date +%Y%m%d).json" \
       "$SONDE/api/v1/config/export"
   )
   ```

2. Stockez le fichier dans un emplacement sauvegardé et à accès restreint.

Fréquence recommandée : après chaque série de modifications importantes, et au minimum une fois par mois.

> **Avertissement** — Un export contenant des secrets chiffrés n'est protégé que par la robustesse de sa phrase de passe. Choisissez une phrase longue, conservez-la séparément du fichier, par exemple dans le coffre-fort de mots de passe de l'équipe.

9.12.3 Restaurer ou dupliquer une configuration

L'import permet de :

- restaurer une configuration sur une sonde réinstallée (chapitre 20) ;
- dupliquer une configuration vers une nouvelle sonde ;
- déployer une configuration de référence sur plusieurs sondes.

Étapes :

1. **Validez** d'abord l'import sans l'appliquer :

   ```bash
   sapi -X POST -H 'Content-Type: application/json' \
     --data-binary @sonde-paris-01-config-20261119.json \
     "$SONDE/api/v1/config/import?dryRun=true"
   ```

L'archive importée a le même format JSON que l'export. Si elle contient des comptes et des secrets chiffrés, transmettez la phrase de passe dans l'en-tête `X-Config-Passphrase`, selon la méthode du § 9.12.2 : reprenez la commande d'export en remplaçant l'appel à `config/export` par la commande d'import ci-dessus, complétée de l'option `-H @"$ENTETE"`. Comme à l'export, la phrase de passe ne doit jamais figurer dans l'adresse, qui peut être journalisée. Pour appliquer la configuration à une sonde qui fonctionnera en parallèle de la sonde d'origine, ajoutez le paramètre `newIdentity=true` (§ 20.5.3). La réponse est un compte rendu JSON qui liste les objets qui seraient créés, modifiés ou supprimés, ainsi que les éventuelles erreurs (fonction absente de l'édition, secret manquant…).

2. Si le rapport est conforme, relancez la même commande sans `?dryRun=true`.
3. Si les secrets n'étaient pas inclus, recréez-les (§ 9.8.2). Tant qu'un secret référencé par un contrôle est absent de la sonde, chaque exécution de ce contrôle se termine en `UNKNOWN` avec le type d'erreur `secret_missing` (annexe H). Si l'archive est antérieure à la liste `allowedHosts`, définissez la liste de chaque secret signalé par le compte rendu d'import avant de compter sur les contrôles qui l'utilisent (§ 9.8.1).
4. Vérifiez par quelques exécutions d'essai que les contrôles fonctionnent.

> **Attention** — L'export n'inclut pas la section `network` (nom d'hôte, adresse IP, DNS) : la sonde cible conserve ses propres paramètres réseau. Lors d'une duplication, vérifiez en revanche les paramètres propres à chaque site (rattachement, attributs de ressource), et utilisez `newIdentity=true` pour que les deux sondes restent distinctes dans les plateformes d'observabilité.

9.13 Automatisation

9.13.1 Principe de la configuration en tant que code

Pour un parc de plusieurs sondes, ou pour une configuration soumise à revue, il est recommandé de gérer les définitions de contrôles comme du **code** :

1. les fichiers JSON des contrôles et des plans sont conservés dans un dépôt Git ;
2. toute modification fait l'objet d'une revue ;
3. un outil d'automatisation applique les définitions sur les sondes par l'API.

Cette approche garantit la traçabilité, la reproductibilité et le retour arrière.

9.13.2 Exemple avec un script shell

Le script suivant, illustratif, crée ou remplace tous les contrôles décrits dans un répertoire `controles/` (un fichier JSON par contrôle). Il s'appuie sur la préparation du § 9.3.3 (variables `SONDE` et `CA`, fichier d'en-tête `entete-auth`), mais définit sa propre fonction d'appel : contrairement à `sapi`, elle récupère le code HTTP de chaque réponse pour décider de la suite.

```bash
#!/usr/bin/env bash
# Déploiement idempotent des contrôles d'un répertoire sur une sonde (exemple)
# Prérequis (§ 9.3.3) : variables SONDE et CA définies, fichier "$HOME/.sva/entete-auth".
set -euo pipefail
shopt -s nullglob                    # motif sans correspondance => liste vide

fichiers=(controles/*.json)
if [ "${#fichiers[@]}" -eq 0 ]; then
  echo "Aucun fichier controles/*.json : rien à déployer." >&2
  exit 1
fi
```

```
tmp=$(mktemp -d)
trap 'rm -rf "$tmp"' EXIT

# Appel de l'API. Positionne deux variables, à tester dans cet ordre :
#   rc   : statut de sortie de curl (0 = transfert complet ; sinon transfert
#          interrompu, erreur TLS, sonde injoignable ou erreur d'écriture locale) ;
#   code : code HTTP reçu (000 si aucun). Un code peut avoir été reçu avant
#          l'interruption du transfert : il n'est fiable que si rc vaut 0.
# Les en-têtes et le corps de la réponse sont conservés dans $tmp.
appel() {
  rm -f "$tmp/entetes" "$tmp/corps"
  rc=0
  code=$(curl -sS --cacert "$CA" -H @"$HOME/.sva/entete-auth" \
              -H 'Accept: application/json' \
              -D "$tmp/entetes" -o "$tmp/corps" -w '%{http_code}' "$@") || rc=$?
}

# Arrêt du déploiement avec le détail renvoyé par la sonde
echec() {
  echo "ERREUR : $1" >&2
  [ -s "$tmp/corps" ] && cat "$tmp/corps" >&2
  exit 1
}

for f in "${fichiers[@]}"; do
  id=$(jq -er '.id' "$f") || echec "$f : identifiant (.id) absent ou JSON invalide"

  # 1. Lecture : transfert d'abord, puis 200 = le contrôle existe, 404 = il n'existe pas
  appel "$SONDE/api/v1/checks/$id"
  [ "$rc" -eq 0 ] ||
    echec "$id : lecture inachevée (curl $rc, HTTP $code) : sonde injoignable, TLS refusé ou
    transfert interrompu ; rien n'a été écrit"
  case "$code" in
    200)
      etag=$(awk 'tolower($1)=="etag:" {print $2}' "$tmp/entetes" | tr -d '\r')
      [ -n "$etag" ] || echec "$id : lecture réussie mais en-tête ETag absent"
      echo "Mise à jour : $id"
      appel -X PUT -H 'Content-Type: application/json' \
            -H "If-Match: $etag" --data-binary @"$f" \
            "$SONDE/api/v1/checks/$id"
      ;;
    404)
      echo "Création   : $id"
      appel -X POST -H 'Content-Type: application/json' \
            --data-binary @"$f" "$SONDE/api/v1/checks"
      ;;
    *) echec "$id : lecture refusée ou impossible (HTTP $code)" ;;
  esac

  # 2. Résultat de l'écriture : transfert d'abord, code HTTP ensuite.
  # Transfert inachevé : l'écriture a pu être appliquée ou non. Arrêt sans
  # réessai ; relisez le contrôle avant de relancer (voir ci-dessous).
  [ "$rc" -eq 0 ] ||
    echec "$id : résultat de l'écriture indéterminé (curl $rc, HTTP $code) ; relisez le contrôle
    avant tout nouvel essai"
  case "$code" in
    2??) ;;                                  # 200, 201 ou 204 : succès
    409) echec "$id : conflit (409), voir la conduite à tenir ci-dessous" ;;
    412) echec "$id : modifié depuis la lecture (412) ; relancez le script" ;;
    *)   echec "$id : écriture refusée (HTTP $code)" ;;
  esac
done
```

Le script distingue explicitement les réponses de la sonde, au lieu de déduire l'absence d'un contrôle d'un échec quelconque de la lecture :

Réponse à la lecture (`GET`)	Conduite du script
`200 OK`	Le contrôle existe. L'en-tête `ETag` est **exigé** ; il est renvoyé dans `If-Match` avec le `PUT` (§ 9.3.5). Son absence arrête le script
`404 Not Found`	Le contrôle n'existe pas : il est créé par `POST`
Transfert inachevé (statut de sortie de `curl` non nul : sonde injoignable, certificat refusé, transfert interrompu, erreur d'écriture locale), **même si un code HTTP a été reçu**	Arrêt immédiat. Le code reçu n'est pas exploité : un `200` ou un `404` suivi d'une interruption ne prouve rien. Aucune écriture n'est tentée
`401`, `403`, `429`, `5xx` ou tout autre code, transfert complet	Arrêt immédiat, avec le détail renvoyé par la sonde. Aucune création n'est tentée
Aucun fichier `controles/*.json`	Arrêt avec un message : grâce à l'option `nullglob`, le motif n'est jamais traité comme un nom de fichier littéral

Pour l'écriture (`PUT` ou `POST`), le script examine aussi le transfert **avant** le code HTTP. Si le transfert est inachevé, le résultat est **indéterminé** : la sonde a pu appliquer l'écriture sans que la réponse soit reçue en entier, ou ne pas l'avoir reçue. Le script s'arrête alors sans réessai et sans compter l'écriture comme réussie. Avant de relancer, relisez le contrôle (`GET /api/v1/checks/{id}`) et comparez-le au fichier du dépôt, ou consultez le journal d'audit (§ 9.10) : vous saurez ainsi si l'écriture a eu lieu. La relance du script reste sûre dans les deux cas, puisqu'elle relit le contrôle avant d'écrire.

Conduite à tenir après un refus de l'écriture :

- **`409 Conflict`** : l'opération est incompatible avec l'état de la sonde. Lors d'une création, l'identifiant a pu être créé entre la lecture et l'écriture, par une autre personne ou un autre outil : relancez le script, qui empruntera alors le chemin de mise à jour. Si le conflit persiste, lisez le détail (`application/problem+json`) : il désigne l'objet en cause, par exemple une référence incompatible avec l'état actuel de la configuration. Corrigez la définition dans le dépôt avant de relancer ;
- **`412 Precondition Failed`** : le contrôle a été modifié entre la lecture et l'écriture. Consultez le journal d'audit (§ 9.10) pour identifier cette modification. Si elle doit être conservée, reportez-la dans le fichier du dépôt ; sinon, relancez simplement le script, qui relit le contrôle et obtient un nouvel `ETag`. Ne contournez jamais ce refus en supprimant l'en-tête `If-Match` ;
- **`422 Unprocessable Content`** : la définition n'est pas conforme au schéma ; corrigez le fichier.

> ! **Attention** — Le script s'arrête à la première erreur : les contrôles traités avant l'erreur restent déployés. Comme il est idempotent, il suffit de le relancer après correction. Les codes de réponse attendus en cas de succès (`200`, `201`, `204`) sont ceux du § 9.3.4.

9.13.3 Exemple avec Ansible

Avec Ansible, le module standard `ansible.builtin.uri` permet d'appeler l'API. L'exemple ci-dessous, illustratif, crée un contrôle s'il n'existe pas. Le jeton est conservé dans un coffre Ansible (`ansible-vault`) et le certificat de la sonde est vérifié.

```yaml
# Exemple illustratif : création d'un contrôle sur les sondes de l'inventaire
- name: Contrôles de supervision
  hosts: sondes
  gather_facts: false
```

```
vars:
  sonde_url: "https://{{ inventory_hostname }}"
  sonde_ca: "files/ca-sondes.pem"          # certificat de l'AC ou de la sonde
  sonde_token: "{{ vault_sonde_token }}"    # jeton chiffré par ansible-vault
tasks:
  - name: Vérifier si le contrôle existe
    no_log: true                            # la tâche manipule le jeton
    ansible.builtin.uri:
      url: "{{ sonde_url }}/api/v1/checks/portail-accueil"
      headers: { Authorization: "Bearer {{ sonde_token }}" }
      ca_path: "{{ sonde_ca }}"
      status_code: [200, 404]
    delegate_to: localhost
    register: existant

  - name: Créer le contrôle
    no_log: true                            # la tâche manipule le jeton
    ansible.builtin.uri:
      url: "{{ sonde_url }}/api/v1/checks"
      method: POST
      headers: { Authorization: "Bearer {{ sonde_token }}" }
      ca_path: "{{ sonde_ca }}"
      body_format: json
      body: "{{ lookup('ansible.builtin.file', 'controles/portail-accueil.json') | from_json }}"
      status_code: 201
    delegate_to: localhost
    when: existant.status == 404
```

Les appels sont délégués au poste de contrôle Ansible (`delegate_to: localhost`) : la sonde est administrée par son API, et non par SSH, qui reste désactivé.

> **Attention** — Le jeton est chiffré dans le coffre Ansible, mais il est déchiffré en mémoire pour construire l'en-tête `Authorization`. Sans précaution, Ansible peut l'afficher en clair dans ses sorties : résultat d'une tâche en échec, mode verbeux (`-v` à `-vvvv`), variable enregistrée par `register` et affichée par `debug`, journal configuré par `log_path`, ou greffon de rappel (*callback*) qui transmet les résultats à un outil tiers. C'est pourquoi **chaque tâche qui manipule le jeton porte `no_log: true`**. Cette option masque aussi le détail des erreurs : pour diagnostiquer un échec, rejouez l'appel avec `curl` selon la méthode du § 9.3.3, plutôt que de retirer `no_log`. N'affichez jamais la variable `existant` avec le module `debug`.

9.13.4 Règles pour une automatisation sûre

Règle	Raison
Utiliser un jeton dédié, de rôle `operator` si possible	Limiter les conséquences d'une fuite du jeton
Vérifier le certificat de la sonde	Éviter l'interception du jeton
Ne jamais placer le jeton dans les arguments d'une commande : fichier d'en-tête protégé (§ 9.3.3), `no_log: true` dans Ansible (§ 9.13.3)	Éviter sa divulgation dans la liste des processus, l'historique ou les journaux
Respecter `If-Match`	Ne pas écraser une modification manuelle concurrente
Valider avec `dryRun` avant d'appliquer à grande échelle	Détecter les erreurs avant leur déploiement
Limiter le rythme des appels	Éviter les réponses `429`
Conserver les secrets hors du dépôt	Les créer par un canal séparé (§ 9.8)

9.14 Gestion centralisée de plusieurs sondes

La console centrale est une évolution prévue et facultative du produit (chapitre 1, § 1.4.4) : elle n'est pas disponible dans la version actuelle, et aucune fonction décrite dans cette documentation n'en dépend. Cette section en présente le principe pour vous permettre d'anticiper son intégration.

> → **Évolution prévue** — Une **console centrale** facultative permettra, dans une version ultérieure, de gérer un parc de sondes depuis un point unique : inventaire, déploiement de configurations, vue consolidée des états. Elle sera proposée au choix en service hébergé (SaaS), à l'adresse réservée `https://fleet-synovex.incitypark.fr`, ou installée sur l'infrastructure du client.

Les principes retenus pour cette console sont les suivants :

Principe	Conséquence
La console est **facultative**	Une sonde sans console fonctionne exactement comme décrit dans cette documentation
La sonde **initie la connexion**, en HTTPS sortant, avec authentification mutuelle par certificat (mTLS)	Aucun port entrant à ouvrir depuis Internet ; compatibilité avec les proxys d'entreprise
La sonde **récupère périodiquement** sa configuration de référence et la valide avant de l'appliquer	Une configuration invalide est rejetée ; la précédente reste en vigueur
La console injoignable **n'interrompt pas** la supervision	La sonde continue avec la dernière configuration appliquée
Les résultats détaillés **restent sur la sonde** et dans vos plateformes d'observabilité	La console ne reçoit qu'une synthèse

D'ici là, la gestion d'un parc repose sur l'API de chaque sonde et sur vos outils d'automatisation (§ 9.13).

Chapitre 10 — Contrôles réseau et techniques

Ce chapitre décrit les cinq types de contrôles qui vérifient la disponibilité et la performance des **couches techniques** : joignabilité réseau, ouverture de ports, résolution de noms, réponses des serveurs web et plugins compatibles avec l'écosystème Nagios. Ce sont les contrôles les plus légers de la sonde. Ils constituent le socle de toute supervision, y compris lorsque des contrôles applicatifs (chapitre 11) ou des scénarios navigateur (partie IV) sont déployés.

Les principes communs à tous les contrôles (fréquence, timeout, nouvelles tentatives, seuils, états) sont expliqués au chapitre 8. La planification et les plans d'exécution sont traités au chapitre 12, la lecture des résultats au chapitre 17. Le présent chapitre se concentre sur ce qui est propre à chaque type de contrôle.

10.1 Vue d'ensemble

10.1.1 Types de contrôles techniques

Type	Ce qui est vérifié	Édition minimale	Composant exécutant
`icmp`	Un équipement répond au ping ; latence et taux de perte	Essential	Environnement d'exécution isolé (utilitaire `fping`)
`tcp`	Un port TCP accepte les connexions ; dialogue simple ; poignée de main TLS	Essential	Moteur principal
`dns`	Un nom est résolu, par le serveur attendu, avec la valeur attendue	Essential	Moteur principal
`http`	Un serveur web ou une API répond avec le code, les en-têtes, le contenu et le délai attendus ; validité du certificat	Essential	Moteur principal
`nagios`	Toute vérification réalisable par un plugin au format Nagios	Essential (plugins fournis) · Professional (plugins personnalisés)	Environnement d'exécution isolé

Le contrôle `http` couvre aussi la supervision d'API et de web services. Ces usages, qui relèvent de la supervision applicative, sont détaillés au chapitre 11.

10.1.2 Conventions des exemples

Tous les exemples de ce chapitre sont **fictifs** :

- noms de domaine en `example.com`, `example.org` et `example.internal` ;
- adresses IP dans les plages réservées à la documentation (`192.0.2.0/24`, `198.51.100.0/24`, `203.0.113.0/24` en IPv4, `2001:db8::/32` en IPv6).

Remplacez-les par les valeurs de votre environnement.

Les contrôles sont présentés au format JSON de l'API d'administration, c'est-à-dire la forme qu'ils prennent dans `POST /api/v1/checks` (chapitre 9). L'interface d'administration propose les mêmes paramètres sous forme de formulaire. Chaque exemple est complet et peut être soumis tel quel, après adaptation des cibles.

> ✓ **Conseil** — Avant d'activer un nouveau contrôle, exécutez-le une première fois en mode essai : `POST /api/v1/checks/{id}/runs?dryRun=true`, ou le bouton d'exécution d'essai de l'interface. Le mode essai réalise le contrôle sans enregistrer d'état ni transmettre de télémétrie. Vous vérifiez ainsi les paramètres sans déclencher d'alerte (chapitre 9, § 9.6.5).

Les extraits de résultats présentés dans ce chapitre illustrent la structure d'un résultat d'exécution (chapitre 17). Les noms des mesures (`measurements`) propres à chaque type de contrôle, tels qu'ils figurent dans ces extraits, sont ceux du produit. Les blocs « Résultat attendu » comportent tous les champs obligatoires du résultat d'exécution (`id`, `checkId`, `checkRevision`, `status`, `startedAt`, `durationMs`) ; les champs facultatifs sans intérêt pour le type de contrôle présenté (étapes, artefacts, trace…) y sont omis. Les blocs signalés comme **extraits (projection)** ne reprennent que quelques champs : ils ne constituent pas une réponse complète de l'API. L'exemple de réponse intégrale figure au chapitre 17, § 17.4.

10.1.3 Cibles IPv6

La sonde fonctionne toujours en IPv4 et peut, en complément, utiliser IPv6 lorsque celui-ci est activé dans ses paramètres réseau (section `network`, objet `ipv6`, chapitre 7). Il est désactivé par défaut. Les contrôles peuvent alors viser des cibles IPv6.

La famille d'adresses employée par un contrôle est fixée par le paramètre commun `ipFamily` (chapitre 8, § 8.1.3) :

- `auto` (défaut) : IPv6 si l'IPv6 est activé sur la sonde et si la cible possède une adresse AAAA, IPv4 sinon ;
- `ipv4` : IPv4 uniquement ;
- `ipv6` : IPv6 uniquement.

Lorsque la cible est désignée par une adresse IP littérale, la famille est celle de cette adresse. Lorsqu'elle est désignée par un nom et que ce nom n'a pas d'adresse dans la famille demandée, l'exécution échoue avec le type d'erreur `dns_error`. Un contrôle `ipFamily: "ipv6"` est refusé à l'enregistrement (`422`) tant que l'IPv6 est désactivé sur la sonde. Si l'IPv6 est désactivé après coup, ces contrôles passent en `UNKNOWN` jusqu'à sa réactivation ou leur modification.

Type	Comportement en IPv6
`icmp`	`fping` envoie des requêtes *ICMPv6 Echo*. Les équipements et pare-feu intermédiaires doivent autoriser l'ICMPv6 Echo depuis l'adresse IPv6 de la sonde
`tcp`	Connexion, poignée de main TLS et dialogue `send` / `expect` identiques à IPv4, vers l'adresse IPv6 retenue
`dns`	Le type d'enregistrement interrogé reste fixé par `dns.recordType` : `AAAA` vérifie les adresses IPv6 d'un nom, quelle que soit la famille utilisée pour joindre le serveur. Les serveurs de `dns.servers` peuvent être des adresses IPv4 ou IPv6 ; chacun est interrogé dans la famille de son adresse
`http`	Requête vers l'adresse IPv6 retenue. Une adresse littérale s'écrit entre crochets dans l'URL : `https://[2001:db8:20::10]/`. La vérification du certificat porte alors sur cette adresse, qui doit figurer dans le certificat du serveur
`nagios`	Selon les options propres à chaque plugin (par exemple `-6` pour la plupart des plugins de **Monitoring Plugins**)

L'exemple fictif suivant vérifie la joignabilité d'un routeur par son adresse IPv6 :

```json
{
  "id": "routeur-site-lyon-ipv6",
  "name": "Routeur du site de Lyon : joignabilité IPv6",
  "description": "Ping ICMPv6 vers l'adresse IPv6 du routeur de Lyon (exemple fictif).",
  "type": "icmp",
  "tags": { "site": "lyon", "domaine": "reseau" },
  "schedule": { "interval": "1m" },
  "timeout": "10s",
  "ipFamily": "ipv6",
  "icmp": {
    "host": "2001:db8:10::1",
    "count": 5,
    "maxPacketLossPercent": 20
  }
}
```

Avec une adresse littérale, `ipFamily` est facultatif : il est indiqué ici pour rendre l'intention explicite. Pour une cible désignée par un nom qui possède des adresses IPv4 et IPv6, `"ipFamily": "ipv6"` force la vérification du chemin IPv6, et un second contrôle en `"ipFamily": "ipv4"` permet de suivre les deux chemins séparément.

Les contrôles vers des cibles IPv6 fonctionnent selon les règles ci-dessus, pour chacun des types du tableau.

10.2 Choisir le bon contrôle

10.2.1 Arbre de décision

Le choix d'un type de contrôle dépend de la question à laquelle vous voulez répondre. Le schéma suivant oriente ce choix. Il inclut les types applicatifs, pour montrer la frontière avec le chapitre 11 et la partie IV.

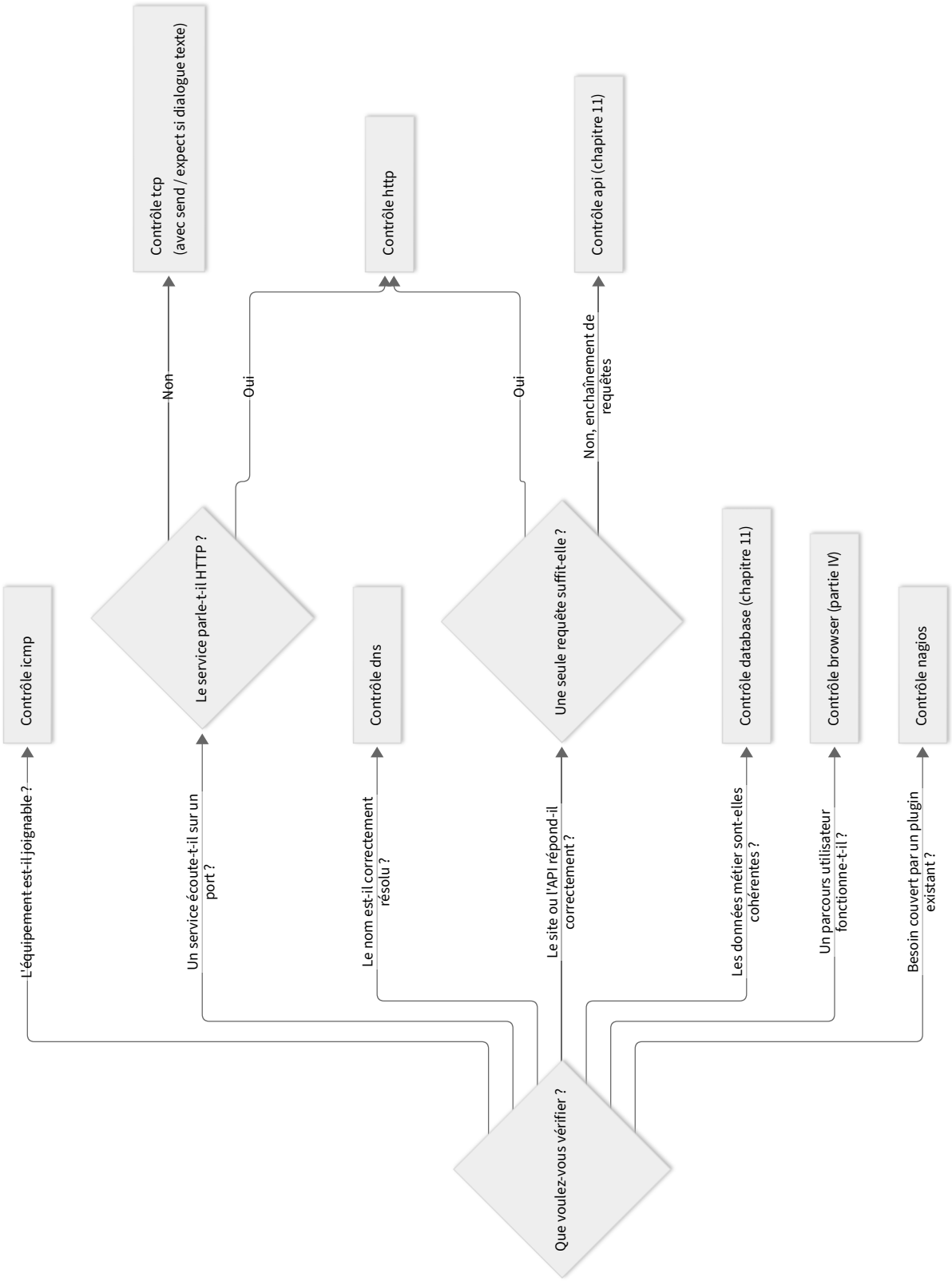


Schéma 10-1 — Choisir un type de contrôle

Schéma présenté en orientation paysage : tournez le document d'un quart de tour pour le lire.

10.2.2 Superviser par couches

Un service informatique repose sur plusieurs couches. Un contrôle unique ne permet pas toujours de localiser une panne : un contrôle HTTP en échec peut révéler un problème de résolution de noms, de réseau, de serveur ou d'application. Superposer des contrôles de niveaux différents sur un même service accélère considérablement le diagnostic.

Exemple pour le site web fictif `www.example.com` :

Couche	Contrôle	Ce que son échec indique
Résolution de noms	`dns` sur `www.example.com`	Le nom n'est plus résolu, ou pointe vers une mauvaise adresse
Réseau	`icmp` sur la passerelle ou le répartiteur de charge	Le chemin réseau est coupé
Transport	`tcp` sur le port 443	Le service n'écoute plus, ou un pare-feu bloque le flux
Protocole	`http` sur la page d'accueil	Le serveur web répond mal (erreur 5xx, contenu inattendu, lenteur)
Certificat	`http` avec surveillance de l'expiration	Le certificat expire bientôt ou est invalide
Application	`browser` sur le parcours de connexion (Enterprise)	L'application est inutilisable pour l'utilisateur

Lors d'un incident, l'état combiné de ces contrôles désigne directement la couche défaillante. Si `dns` et `tcp` sont `OK` mais `http` est `CRITICAL`, la cause se situe au niveau du serveur web ou de l'application, pas du réseau.

> ✓ **Conseil** — N'empilez pas systématiquement toutes les couches pour chaque service. Réservez cette approche aux services critiques. Pour les autres, un contrôle `http` bien configuré suffit souvent.

10.3 Contrôle ICMP (`icmp`)

10.3.1 Objectif et intérêt

Le contrôle `icmp` envoie une série de requêtes *ICMP Echo* (le « ping ») vers un équipement et mesure :

- le **temps d'aller-retour** (RTT, *Round-Trip Time*) : délai entre l'envoi d'une requête et la réception de la réponse ;
- le **taux de perte** : proportion de requêtes restées sans réponse.

Il répond à la question la plus élémentaire de la supervision : l'équipement est-il joignable depuis la sonde ? Il convient particulièrement aux équipements qui n'exposent aucun service applicatif interrogeable : routeurs, commutateurs, pare-feu, liens inter-sites, imprimantes, équipements industriels.

Le contrôle `icmp` ne vérifie **pas** qu'un service fonctionne. Un serveur peut répondre au ping alors que son application est arrêtée. À l'inverse, de nombreux serveurs et pare-feu bloquent volontairement l'ICMP : un échec ne signifie pas toujours une panne.

10.3.2 Fonctionnement

Le contrôle est exécuté par l'environnement d'exécution isolé de la sonde, au moyen de l'utilitaire système `fping`. Pour chaque exécution, la sonde :

1. résout le nom de l'hôte s'il ne s'agit pas d'une adresse IP, en IPv4 ou en IPv6 selon le paramètre `ipFamily` (§ 10.1.3) ;
2. envoie le nombre de requêtes défini par `count` ;
3. calcule le temps d'aller-retour moyen et le taux de perte ;
4. compare ces valeurs aux seuils.

10.3.3 Paramètres

Paramètre	Obligatoire	Description	Défaut
`icmp.host`	Oui	Nom d'hôte ou adresse IP (IPv4 ou IPv6) de l'équipement	—
`icmp.count`	Non	Nombre de requêtes par exécution (1 à 20)	3
`icmp.maxPacketLossPercent`	Non	Taux de perte maximal toléré, en pourcentage. Au-delà, l'état est `CRITICAL`	0
`icmp.warningRttMs`	Non	Temps d'aller-retour moyen, en millisecondes, au-delà duquel l'état est `WARNING`	Aucun
`icmp.criticalRttMs`	Non	Temps d'aller-retour moyen, en millisecondes, au-delà duquel l'état est `CRITICAL`	Aucun

Les paramètres communs (`schedule`, `timeout`, `retry`, `tags`…) sont décrits au chapitre 8.

10.3.4 Détermination de l'état

Situation	État	Type d'erreur
Taux de perte inférieur ou égal au seuil et RTT moyen sous `warningRttMs`	`OK`	—
Taux de perte inférieur ou égal au seuil, mais RTT moyen supérieur à `warningRttMs`	`WARNING`	—
Taux de perte inférieur ou égal au seuil, mais RTT moyen supérieur à `criticalRttMs`	`CRITICAL`	—
Taux de perte supérieur à `maxPacketLossPercent`, y compris aucune réponse	`CRITICAL`	`timeout`
Nom d'hôte non résolu	`CRITICAL`	`dns_error`
Utilitaire indisponible ou erreur interne de la sonde	`UNKNOWN`	`plugin_error`

10.3.5 Exemple : joignabilité d'un routeur de site distant

Le routeur fictif du site de Lyon, d'adresse `192.0.2.1`, est contrôlé toutes les minutes. Cinq requêtes sont envoyées par exécution :

- jusqu'à une requête perdue sur cinq (20 %) est tolérée ;
- un RTT moyen supérieur à 50 ms déclenche un `WARNING`.

```json
{
  "id": "routeur-site-lyon",
  "name": "Routeur du site de Lyon : joignabilité",
  "description": "Vérifie la liaison inter-sites vers le routeur de Lyon (exemple fictif).",
  "type": "icmp",
  "tags": { "site": "lyon", "domaine": "reseau" },
  "schedule": { "interval": "1m" },
  "timeout": "10s",
  "retry": { "attempts": 2, "interval": "15s" },
  "icmp": {
    "host": "192.0.2.1",
    "count": 5,
    "maxPacketLossPercent": 20,
    "warningRttMs": 50
  }
}
```

Les deux nouvelles tentatives à 15 secondes d'intervalle évitent qu'une perte ponctuelle sur la liaison ne déclenche une alerte. L'état `CRITICAL` n'est retenu que si les trois exécutions successives échouent (chapitre 8).

10.3.6 Résultat attendu

Exemple de résultat d'exécution réussie :

```json
{
  "id": "01M7SD3NCWRFNSFJGQGVKWDBH3",
  "checkId": "routeur-site-lyon",
  "checkRevision": 1,
  "status": "OK",
  "startedAt": "2025-11-19T08:00:00.412Z",
  "durationMs": 4120,
  "measurements": {
    "icmp.rtt_avg_ms": 12.4,
    "icmp.packet_loss_percent": 0
  }
}
```

Ces deux mesures alimentent les métriques `sva.icmp.rtt` et `sva.icmp.packet_loss` (chapitre 18).

10.3.7 Erreurs courantes

Symptôme	Cause probable	Action
`CRITICAL` permanent alors que l'équipement fonctionne	L'équipement ou un pare-feu intermédiaire bloque les requêtes ICMP	Faire autoriser l'ICMP Echo depuis l'adresse de la sonde, ou remplacer le contrôle par un contrôle `tcp` sur un port ouvert
États alternant entre `OK` et `CRITICAL`	Liaison instable, ou limitation du débit ICMP par l'équipement (les routeurs donnent souvent une faible priorité à l'ICMP)	Augmenter `count` et `maxPacketLossPercent` ; configurer des nouvelles tentatives
`WARNING` fréquents sans impact pour les utilisateurs	Seuil `warningRttMs` trop strict pour une liaison longue distance	Mesurer la latence normale pendant quelques jours, puis fixer le seuil avec une marge
`CRITICAL` avec `dns_error`	Le nom de l'équipement n'est pas résolu par les serveurs DNS de la sonde	Vérifier la configuration DNS de la sonde (chapitre 7) ou utiliser l'adresse IP

> **Information** — Le temps d'aller-retour ICMP mesure la latence réseau, pas le temps de réponse d'une application. Pour suivre la performance perçue par les utilisateurs, utilisez les mesures du contrôle `http` (§ 10.6) ou d'un scénario navigateur.

10.4 Contrôle TCP (`tcp`)

10.4.1 Objectif et intérêt

Le contrôle `tcp` vérifie qu'un **port TCP** accepte les connexions sur un hôte donné, et mesure le temps d'établissement de la connexion. Il permet en outre :

- d'effectuer une **poignée de main TLS** (*TLS handshake*), la négociation initiale d'une connexion chiffrée, pour vérifier qu'un service chiffré répond correctement ;
- de mener un **dialogue texte simple** : envoyer une chaîne de caractères (`send`), puis vérifier que la réponse contient une chaîne attendue (`expect`).

C'est le contrôle de référence pour les services qui ne parlent pas HTTP : serveurs de messagerie (SMTP, IMAP), annuaires (LDAP), bases de données (au niveau réseau), serveurs de fichiers, files de messages, caches, protocoles propriétaires.

Il est plus fiable que le ping pour vérifier qu'un serveur est opérationnel, puisqu'il atteste qu'un processus écoute effectivement sur le port. Il ne valide en revanche ni l'authentification, ni le bon fonctionnement applicatif du service.

10.4.2 Paramètres

Paramètre	Obligatoire	Description	Défaut
`tcp.host`	Oui	Nom d'hôte ou adresse IP (IPv4 ou IPv6, § 10.1.3)	—
`tcp.port`	Oui	Port TCP (1 à 65535)	—
`tcp.tls`	Non	Si `true`, effectue une poignée de main TLS après la connexion	`false`
`tcp.send`	Non	Chaîne envoyée après l'établissement de la connexion (1 024 caractères maximum). Les séquences JSON `\r` et `\n` permettent d'envoyer des retours chariot et des sauts de ligne	Aucune
`tcp.expect`	Non	Chaîne qui doit figurer dans les données reçues (1 024 caractères maximum)	Aucune
`thresholds.warningDuration`	Non	Durée totale au-delà de laquelle l'état est `WARNING`	Aucun

> **Information** — La validation du certificat lors de la poignée de main TLS du contrôle `tcp` (chaîne de confiance, nom d'hôte) suit le magasin de certificats de la sonde : le magasin système, complété des autorités de certification importées dans la section `trust` (§ 7.7.3). Pour une surveillance détaillée d'un certificat, y compris de sa date d'expiration, utilisez un contrôle `http` (§ 10.6.7) lorsque le service est un service web.

10.4.3 Détermination de l'état

Situation	État	Type d'erreur
Connexion établie, poignée de main TLS réussie si demandée, chaîne attendue reçue si définie, durée sous le seuil	`OK`	—

Situation	État	Type d'erreur
Idem, mais durée supérieure à `thresholds.warningDuration`	`WARNING`	—
Connexion refusée (aucun processus n'écoute, ou rejet actif par un pare-feu)	`CRITICAL`	`connection_refused`
Pas de réponse dans le délai imparti (filtrage silencieux, hôte éteint)	`CRITICAL`	`timeout`
Connexion interrompue par le serveur	`CRITICAL`	`connection_reset`
Échec de la poignée de main TLS	`CRITICAL`	`tls_error`
Chaîne `expect` absente de la réponse	`CRITICAL`	`assertion_failed`
Nom d'hôte non résolu	`CRITICAL`	`dns_error`

10.4.4 Exemple : port d'un serveur de base de données

Ce contrôle vérifie que le serveur PostgreSQL fictif `db-ventes.example.internal` accepte les connexions sur son port standard. Il ne remplace pas un contrôle `database` (chapitre 11), qui exécute une requête réelle. Il suffit en revanche à distinguer une panne réseau d'une panne applicative.

```json
{
  "id": "db-ventes-port",
  "name": "Serveur de base Ventes : port PostgreSQL",
  "type": "tcp",
  "tags": { "application": "ventes", "domaine": "base_de_donnees" },
  "schedule": { "interval": "1m" },
  "timeout": "5s",
  "retry": { "attempts": 1, "interval": "20s" },
  "thresholds": { "warningDuration": "500ms" },
  "tcp": {
    "host": "db-ventes.example.internal",
    "port": 5432
  }
}
```

10.4.5 Exemple : bannière d'un serveur de messagerie

Un serveur SMTP envoie une bannière commençant par le code `220` dès l'ouverture de la connexion. Vérifier cette bannière garantit que le service de messagerie est prêt à recevoir des messages, et pas seulement que le port est ouvert.

```json
{
  "id": "smtp-relais",
  "name": "Relais SMTP interne : bannière de service",
  "type": "tcp",
  "tags": { "application": "messagerie" },
  "schedule": { "interval": "2m" },
  "timeout": "10s",
  "tcp": {
    "host": "smtp.example.internal",
    "port": 25,
    "expect": "220"
  }
}
```

Aucune chaîne n'est envoyée : la sonde se contente de lire la bannière. Le contrôle ne transmet aucun message et ne s'authentifie pas.

10.4.6 Exemple : dialogue texte avec un cache

Certains services acceptent des commandes texte. L'exemple suivant envoie la commande `PING` à un serveur de cache compatible avec le protocole Redis, qui doit répondre `+PONG`. La séquence `\r\n` termine la commande, conformément au protocole.

JSON

```
{
  "id": "cache-sessions",
  "name": "Cache des sessions : réponse au PING",
  "type": "tcp",
  "tags": { "application": "portail" },
  "schedule": { "interval": "1m" },
  "timeout": "5s",
  "tcp": {
    "host": "cache.example.internal",
    "port": 6379,
    "send": "PING\r\n",
    "expect": "+PONG"
  }
}
```

> **Attention** — N'envoyez jamais par `send` une commande qui modifie l'état du service : écriture, suppression, arrêt. Le contrôle s'exécute à intervalle régulier, potentiellement des milliers de fois par jour. Limitez-vous aux commandes de lecture ou de diagnostic prévues par le protocole.

10.4.7 Exemple : service chiffré (TLS)

Pour un annuaire LDAP accessible en LDAPS (port 636), l'option `tls` vérifie que la négociation chiffrée aboutit :

JSON

```
{
  "id": "annuaire-ldaps",
  "name": "Annuaire d'entreprise : LDAPS",
  "type": "tcp",
  "tags": { "application": "annuaire" },
  "schedule": { "interval": "2m" },
  "timeout": "10s",
  "tcp": {
    "host": "ldap.example.internal",
    "port": 636,
    "tls": true
  }
}
```

10.4.8 Résultat attendu

JSON

```
{
  "id": "01M7SDCT5TPA021PRK6SRM48FC",
  "checkId": "smtp-relais",
  "checkRevision": 1,
  "status": "OK",
  "startedAt": "2025-11-19T08:05:00.218Z",
  "durationMs": 38,
  "measurements": {
    "tcp.connect_ms": 4,
    "tcp.response_received": true
  }
}
```

La durée totale de l'exécution alimente la métrique `sva.check.duration` (chapitre 18) ; la durée d'établissement de la connexion est conservée dans les mesures du résultat.

10.4.9 Erreurs courantes

Symptôme	Cause probable	Action
`CRITICAL` / `connection_refused`	Le service est arrêté, écoute sur une autre interface ou un autre port, ou un pare-feu rejette la connexion	Vérifier l'état du service sur le serveur ; vérifier le port ; vérifier les règles de filtrage entre la sonde et la cible
`CRITICAL` / `timeout`	Un pare-feu ignore silencieusement les paquets, ou l'hôte est éteint	Faire ouvrir le flux depuis l'adresse de la sonde (chapitre 7) ; tester la joignabilité avec un contrôle `icmp`
`CRITICAL` / `assertion_failed` alors que le service fonctionne	La chaîne attendue diffère (bannière personnalisée, autre version) ou la réponse arrive en plusieurs fragments après la lecture	Exécuter le contrôle en mode essai et lire la réponse reçue dans le détail de l'exécution ; choisir une chaîne `expect` plus courte et stable
`CRITICAL` / `tls_error`	Certificat expiré ou non reconnu, protocole TLS non pris en charge, service non chiffré sur ce port	Vérifier le certificat du service ; vérifier que le port est bien un port TLS (et non STARTTLS, qui commence en clair)
Le contrôle provoque des entrées d'erreur dans les journaux du service	Connexions ouvertes puis fermées sans dialogue complet, considérées comme anormales par certains services	En informer l'équipe responsable ; espacer la fréquence ; préférer un plugin Nagios dédié au protocole (§ 10.7)

> **Information** — Les services qui utilisent STARTTLS (SMTP sur le port 587, IMAP sur 143, LDAP sur 389) débutent la connexion en clair et ne basculent en chiffré qu'après une commande explicite. Le contrôle `tcp` avec `tls: true` ne convient pas à ce cas. Utilisez le plugin Nagios adapté (`check_smtp` avec l'option `-S` , par exemple), décrit au § 10.7.

10.5 Contrôle DNS (`dns`)

10.5.1 Objectif et intérêt

Le DNS (*Domain Name System*) traduit les noms (`www.example.com`) en adresses IP et fournit d'autres informations : serveurs de messagerie d'un domaine, alias, enregistrements texte. Quand le DNS est défaillant, tous les services qui en dépendent deviennent inaccessibles, même s'ils fonctionnent parfaitement.

Le contrôle `dns` interroge un ou plusieurs serveurs DNS et vérifie :

- que le nom est **résolu** (la requête aboutit et renvoie au moins une réponse) ;
- que la réponse **contient les valeurs attendues** (adresse IP, nom de serveur de messagerie…) ;
- que le **délai de résolution** reste acceptable.

Cas d'usage typiques :

- détecter une erreur de modification d'une zone DNS : enregistrement supprimé ou adresse modifiée par erreur ;
- vérifier que les serveurs DNS internes répondent, en interrogeant chaque serveur individuellement ;
- vérifier la cohérence entre la vue interne et la vue externe d'un domaine (*split-horizon*) ;
- contrôler les enregistrements de messagerie (MX) ou les enregistrements texte (TXT) utilisés pour la sécurité des messages ;
- détecter une redirection anormale du trafic par une modification non autorisée d'un enregistrement.

10.5.2 Paramètres

Paramètre	Obligatoire	Description	Défaut
`dns.name`	Oui	Nom à résoudre	—
`dns.recordType`	Non	Type d'enregistrement : `A`, `AAAA`, `CNAME`, `MX`, `TXT`, `NS`, `SRV`, `PTR`, `SOA`	`A`
`dns.servers`	Non	Serveurs DNS à interroger (4 au maximum), désignés par une adresse IPv4 ou IPv6 (§ 10.1.3). S'il est absent, la sonde utilise ses propres serveurs DNS (chapitre 7)	Serveurs de la sonde
`dns.expect`	Non	Valeurs qui doivent figurer dans la réponse (16 au maximum)	Aucune
`thresholds.warningDuration`	Non	Délai de résolution au-delà duquel l'état est `WARNING`	Aucun

Types d'enregistrements :

Type	Contenu	Exemple de valeur attendue
`A`	Adresse IPv4	`203.0.113.10`
`AAAA`	Adresse IPv6	`2001:db8::10`
`CNAME`	Alias vers un autre nom	`www-prod.example.net`
`MX`	Serveurs de messagerie du domaine	`mx1.example.com`
`TXT`	Texte libre (politiques de messagerie, vérifications de domaine)	`v=spf1 include:_spf.example.com -all`
`NS`	Serveurs faisant autorité pour la zone	`ns1.example.com`
`SRV`	Localisation d'un service (hôte et port)	`ldap.example.internal`
`PTR`	Résolution inverse (adresse vers nom)	`www.example.com`
`SOA`	Informations d'autorité de la zone	Serveur primaire de la zone

> **Information** — Lorsque plusieurs valeurs sont listées dans `expect`, toutes doivent être présentes dans la réponse. La réponse peut contenir d'autres valeurs en plus. Lorsque plusieurs serveurs sont listés dans `servers`, chacun est interrogé et chacun doit répondre correctement.

10.5.3 Détermination de l'état

Situation	État	Type d'erreur
Réponse obtenue, valeurs attendues présentes, délai sous le seuil	`OK`	—
Idem, mais délai supérieur à `thresholds.warningDuration`	`WARNING`	—
Nom inexistant (NXDOMAIN), absence d'enregistrement du type demandé, échec du serveur (SERVFAIL)	`CRITICAL`	`dns_error`
Serveur DNS ne répondant pas dans le délai	`CRITICAL`	`timeout`
Réponse obtenue, mais une valeur attendue est absente	`CRITICAL`	`assertion_failed`

10.5.4 Exemple : adresse publique d'un site

Ce contrôle vérifie que le nom fictif `www.example.com` pointe toujours vers l'adresse publique attendue. Il interroge deux résolveurs : l'un interne, l'autre représentant un résolveur externe de l'entreprise.

```json
{
  "id": "dns-site-public",
  "name": "DNS : adresse du site public",
  "type": "dns",
  "tags": { "application": "site_public", "domaine": "dns" },
  "schedule": { "interval": "5m" },
  "timeout": "10s",
  "retry": { "attempts": 1, "interval": "30s" },
  "thresholds": { "warningDuration": "300ms" },
  "dns": {
    "name": "www.example.com",
    "recordType": "A",
    "servers": ["192.0.2.53", "198.51.100.53"],
    "expect": ["203.0.113.10"]
  }
}
```

Si quelqu'un modifie l'enregistrement par erreur, le contrôle passe en `CRITICAL` avec le type d'erreur `assertion_failed`, alors même que le nom continue d'être résolu.

10.5.5 Exemple : enregistrements de messagerie

```json
{
  "id": "dns-mx-domaine",
  "name": "DNS : serveurs de messagerie du domaine",
  "type": "dns",
  "tags": { "application": "messagerie", "domaine": "dns" },
  "schedule": { "interval": "15m" },
  "timeout": "10s",
  "dns": {
    "name": "example.com",
    "recordType": "MX",
    "expect": ["mx1.example.com", "mx2.example.com"]
  }
}
```

10.5.6 Exemple : résolution interne d'un service applicatif

```json
{
  "id": "dns-api-interne",
  "name": "DNS interne : nom de l'API de commandes",
  "type": "dns",
  "tags": { "application": "commandes", "domaine": "dns" },
  "schedule": { "interval": "5m" },
  "timeout": "5s",
  "dns": {
    "name": "api-commandes.example.internal",
    "recordType": "A"
  }
}
```

Sans `expect`, le contrôle vérifie uniquement que le nom est résolu, quelle que soit l'adresse. Cette forme convient aux services dont l'adresse change légitimement (répartition de charge dynamique, plateformes en nuage).

10.5.7 Résultat attendu

```json
{
  "id": "01M7SDNZ8B64S4DFX3Q75QPQA1",
  "checkId": "dns-site-public",
  "checkRevision": 1,
  "status": "OK",
  "startedAt": "2025-11-19T08:10:00.331Z",
  "durationMs": 42,
  "measurements": {
    "dns.answers": "203.0.113.10",
    "dns.lookup_ms": 18
  }
}
```

Le délai de résolution alimente la métrique `sva.dns.lookup.duration` (chapitre 18).

10.5.8 Erreurs courantes

Symptôme	Cause probable	Action
`CRITICAL` / `dns_error` sur un nom interne uniquement	Les serveurs DNS de la sonde ne connaissent pas la zone interne (sonde configurée avec des résolveurs publics)	Configurer les serveurs DNS internes sur la sonde (chapitre 7) ou dans `dns.servers`
`CRITICAL` / `assertion_failed` alors que le service fonctionne	L'adresse a été modifiée volontairement (migration, changement d'hébergeur), ou plusieurs adresses sont renvoyées en rotation	Mettre à jour `expect` ; ne lister que les valeurs stables
`CRITICAL` / `timeout` sur un serveur précis	Le serveur DNS est arrêté, ou le flux UDP/TCP 53 est filtré entre la sonde et ce serveur	Vérifier le service DNS ; faire ouvrir le port 53 en UDP et TCP
Résultats différents selon le serveur interrogé	Vue interne et vue externe différentes (comportement voulu), ou réplication incomplète entre serveurs	Créer un contrôle par vue avec les valeurs attendues propres à chacune ; vérifier la réplication des zones
Le contrôle reste `OK` après une modification de zone	Le résolveur interrogé renvoie la valeur conservée en cache jusqu'à expiration de sa durée de vie (TTL)	Interroger directement les serveurs faisant autorité dans `dns.servers`

10.6 Contrôle HTTP et HTTPS (`http`)

10.6.1 Objectif et intérêt

Le contrôle `http` envoie une requête HTTP ou HTTPS et vérifie la réponse : code de statut, en-têtes, contenu, délai. Il mesure aussi les **phases** de la requête, ce qui permet de localiser une lenteur, et surveille le **certificat** des sites HTTPS.

C'est le contrôle le plus utilisé de la sonde. Il convient pour :

- les sites web et les pages d'accueil ;
- les points de contrôle de santé des applications (*health checks*), souvent exposés sous des chemins comme `/health` ou `/status` ;
- les API REST et les web services : voir le chapitre 11 pour les usages applicatifs avancés ;
- la surveillance des certificats TLS ;
- la vérification des redirections : passage forcé en HTTPS, nouvelles adresses après une migration.

Contrairement à un navigateur, le contrôle `http` n'exécute pas le JavaScript des pages et ne charge pas les ressources associées (images, feuilles de style, scripts). Il vérifie une seule requête et sa réponse. Pour vérifier le comportement d'une application web telle qu'un utilisateur la voit, utilisez un scénario navigateur (partie IV).

10.6.2 Paramètres de la requête

Paramètre	Obligatoire	Description	Défaut
`http.url`	Oui	URL complète, `http://` ou `https://` (2 048 caractères maximum). Une adresse IPv6 littérale s'écrit entre crochets (§ 10.1.3)	—
`http.method`	Non	`GET`, `HEAD`, `POST`, `PUT`, `PATCH`, `DELETE`, `OPTIONS`	`GET`
`http.headers`	Non	En-têtes de la requête (32 au maximum). Les valeurs sensibles référencent un secret : `${secret:NOM}`	Aucun
`http.body`	Non	Corps de la requête (64 Kio maximum), pour `POST`, `PUT` et `PATCH`	Aucun
`http.followRedirects`	Non	Suivre les redirections (codes 301, 302, 303, 307, 308)	`true`
`http.tls.verify`	Non	Vérifier le certificat du serveur (chaîne de confiance et nom)	`true`
`http.tls.caRef`	Non	Restreint la confiance de ce contrôle à une seule autorité de certification importée dans la section `trust` (§ 7.7.3). Sans cette propriété, le magasin complet de la sonde s'applique	Aucune
`http.tls.certificateExpiryWarningDays`	Non	Nombre de jours avant l'expiration du certificat en dessous duquel l'état est `WARNING` (0 à 365)	Aucun
`http.tls.certificateExpiryCriticalDays`	Non	Nombre de jours avant l'expiration du certificat en dessous duquel l'état est `CRITICAL` (0 à 365)	Aucun
`http.proxy`	Non	`"none"` : le contrôle contourne le proxy global de la sonde (chapitre 7). C'est la seule valeur admise : un contrôle ne peut pas définir un autre proxy	Proxy global, avec ses exclusions
`http.assertions`	Non	Liste des vérifications à appliquer à la réponse (20 au maximum)	Voir § 10.6.4

Le délai maximal de la requête est fixé par le paramètre commun `timeout` du contrôle (chapitre 8).

10.6.3 Méthodes HTTP

Méthode	Usage en supervision	Précaution
`GET`	Cas général : récupérer une page ou une ressource	Aucune
`HEAD`	Vérifier l'existence d'une ressource et ses en-têtes sans télécharger le contenu : économie de bande passante pour les gros fichiers	Certains serveurs traitent mal `HEAD` ; les assertions sur le corps sont alors impossibles
`POST`	Interroger un service de recherche, un web service SOAP, un point d'authentification	Vérifier que la requête **ne crée ni ne modifie** de données

Méthode	Usage en supervision	Précaution
`PUT`, `PATCH`, `DELETE`	Rarement utiles ; à réserver aux environnements de test	Ces méthodes modifient des données : **ne jamais les utiliser en production**
`OPTIONS`	Vérifier les méthodes autorisées ou la configuration CORS	Aucune

> **Avertissement** — Une requête de supervision est répétée en permanence. Une requête `POST` qui crée une commande, un ticket ou un compte, exécutée toutes les minutes, produit plus de 1 400 enregistrements par jour. N'utilisez que des requêtes sans effet de bord, ou des environnements de test dédiés (chapitre 11, § 11.9).

10.6.4 Assertions

Une **assertion** est une condition que la réponse doit satisfaire. Si au moins une assertion échoue, l'état est `CRITICAL` avec le type d'erreur `assertion_failed` (ou `http_status` pour une assertion sur le code de statut).

Chaque assertion comporte :

- un **type** (`kind`) ;
- un **opérateur** (`operator`) ;
- une **valeur attendue** (`value`) ;
- selon le type, un **chemin** (`path`).

Type (`kind`)	`path`	Ce qui est comparé	Exemple
`status`	—	Code de statut HTTP de la réponse finale	Égal à `200`
`header`	Nom de l'en-tête	Valeur de l'en-tête de réponse	`Content-Type` contient `application/json`
`body`	—	Corps de la réponse, en texte	Contient `Bienvenue`
`jsonPath`	Expression JSONPath	Valeur extraite d'une réponse JSON	`$.status` égal à `UP`
`xpath`	Expression XPath	Valeur extraite d'une réponse XML	`//*[local-name()='etat']` égal à `actif`
`duration`	—	Durée totale de la requête, en millisecondes	Inférieure à `2000`

Opérateurs disponibles :

Opérateur	Signification	Types de valeurs
`eq` / `ne`	Égal / différent	Nombre, texte, booléen
`lt` / `lte`	Strictement inférieur / inférieur ou égal	Nombre
`gt` / `gte`	Strictement supérieur / supérieur ou égal	Nombre
`contains` / `notContains`	Contient / ne contient pas	Texte
`matches`	Correspond à une expression régulière	Texte (expression régulière)
`exists`	L'élément est présent (en-tête, chemin JSON ou XML)	Aucune valeur requise

Une **expression JSONPath** désigne une valeur dans un document JSON : `$` représente la racine, `$.status` le champ `status` de premier niveau, `$.items[0].id` le champ `id` du premier élément du tableau `items`. Une **expression XPath** joue le même rôle pour les documents XML.

> **Information** — Si aucune assertion de type `status` n'est définie, une réponse dont le code est supérieur ou égal à 400 est considérée comme un échec (`CRITICAL`, `http_status`). Définissez toujours une assertion `status` explicite : le comportement du contrôle est alors sans ambiguïté.

10.6.5 Phases mesurées

Pour chaque exécution, la sonde décompose la durée de la requête en phases. Cette décomposition est l'outil principal de diagnostic d'une lenteur.

Phase	Mesure	Une valeur élevée indique…
`dns`	Résolution du nom d'hôte	Un serveur DNS lent ou surchargé
`connect`	Établissement de la connexion TCP	Une latence réseau, une saturation, un pare-feu lent
`tls`	Négociation TLS (HTTPS uniquement)	Un serveur surchargé en calcul cryptographique, une chaîne de certificats longue
`ttfb`	Délai jusqu'au premier octet de réponse (*Time To First Byte*), une fois la requête envoyée	Un traitement lent côté serveur : application, base de données
`transfer`	Réception du corps de la réponse	Une réponse volumineuse ou une bande passante insuffisante

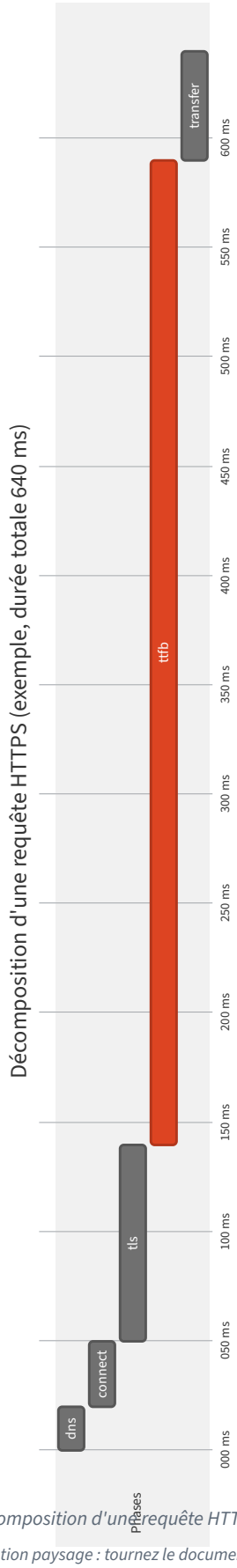


Schéma 10-2 — Décomposition d'une requête HTTPS en phases (exemple)

Schéma présenté en orientation paysage : tournez le document d'un quart de tour pour le lire.

Dans cet exemple, l'essentiel du temps est consommé par la phase `ttfb` : le serveur met 450 ms à produire sa réponse. Le réseau n'est pas en cause. Ces phases sont publiées sous la métrique `sva.http.phase.duration` (chapitre 18).

10.6.6 Redirections

Par défaut (`followRedirects: true`), la sonde suit les redirections, et les assertions portent sur la **réponse finale**. Pour vérifier la redirection elle-même, par exemple pour garantir que le site non chiffré renvoie bien vers HTTPS, désactivez le suivi et vérifiez le code et l'en-tête `Location` (§ 10.6.10).

10.6.7 Certificats et expiration

Pour une URL en `https://`, la sonde vérifie le certificat présenté par le serveur :

Vérification	Échec	État
Le certificat est signé par une autorité de confiance (magasin de la sonde ou `caRef`)	Certificat autosigné ou autorité inconnue	`CRITICAL` / `tls_error`
Le nom du serveur correspond au certificat	Certificat émis pour un autre nom	`CRITICAL` / `tls_error`
Le certificat est en cours de validité	Certificat expiré	`CRITICAL` / `certificate_expired`
L'expiration est au-delà du délai de prévenance	Expiration dans moins de `certificateExpiryWarningDays` jours	`WARNING`
L'expiration est au-delà du délai critique	Expiration dans moins de `certificateExpiryCriticalDays` jours	`CRITICAL`

La durée restante avant expiration est publiée en permanence dans la métrique `sva.tls.certificate.expiry` (en secondes), ce qui permet de construire un tableau de bord de tous les certificats supervisés.

> ✓ **Conseil** — Fixez `certificateExpiryWarningDays` à une valeur compatible avec votre processus de renouvellement. Pour des certificats renouvelés manuellement avec une validation administrative, 30 jours est un minimum. Pour des certificats renouvelés automatiquement, une valeur plus courte (7 à 14 jours) signale le dysfonctionnement du renouvellement automatique sans fausse alerte.

10.6.8 Sites internes et autorité de certification d'entreprise

Les applications internes sont souvent protégées par des certificats émis par l'autorité de certification (AC) de l'entreprise, inconnue du magasin standard de la sonde. Dans ce cas :

1. Importez le certificat de l'AC interne, au format PEM, dans la section `trust` des paramètres de la sonde (§ 7.7.3). Ce magasin s'ajoute au magasin système et s'applique aux contrôles `http`, `api`, `tcp` (avec TLS) et `browser`, ainsi qu'aux exports.
2. Le cas échéant, référencez-la explicitement dans le contrôle avec `http.tls.caRef`.

> ! **Avertissement** — Le paramètre `tls.verify: false` désactive toute vérification du certificat. Le contrôle ne détecte plus alors une usurpation de serveur, ni l'expiration d'un certificat. Réservez cette option à un diagnostic temporaire, jamais à un contrôle permanent.

10.6.9 Exemple : page d'accueil d'un site

Ce contrôle vérifie que la page d'accueil du site fictif répond avec le code 200, contient le texte attendu, ne présente pas de message d'erreur applicatif, répond en moins de 3 secondes et dispose d'un certificat valide au moins 21 jours.

JSON

```
{
  "id": "site-public-accueil",
  "name": "Site public : page d'accueil",
  "type": "http",
  "tags": { "application": "site_public", "environment": "production" },
  "schedule": { "interval": "1m", "jitter": "10s" },
  "timeout": "15s",
  "retry": { "attempts": 1, "interval": "30s" },
  "thresholds": { "warningDuration": "2s" },
  "http": {
    "method": "GET",
    "url": "https://www.example.com/",
    "headers": { "User-Agent": "Supervision-Sonde (exemple)" },
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "body", "operator": "contains", "value": "Bienvenue sur Example" },
      { "kind": "body", "operator": "notContains", "value": "Erreur interne" },
      { "kind": "duration", "operator": "lt", "value": 3000 }
    ],
    "tls": { "verify": true, "certificateExpiryWarningDays": 21 }
  }
}
```

Lecture de cet exemple :

- Au-delà de 2 secondes (`warningDuration`), l'état passe à `WARNING`. Au-delà de 3 secondes (assertion `duration`), il passe à `CRITICAL`.
- L'assertion `notContains` détecte les pages d'erreur renvoyées avec un code 200. Ce cas est fréquent lorsqu'un cadre applicatif intercepte les exceptions.
- L'en-tête `User-Agent` identifie la sonde dans les journaux du serveur web. Les équipes d'exploitation peuvent ainsi distinguer le trafic de supervision de celui des utilisateurs, et l'exclure de leurs statistiques d'audience.

10.6.10 Exemple : redirection vers HTTPS

JSON

```
{
  "id": "site-public-redirection-https",
  "name": "Site public : redirection HTTP vers HTTPS",
  "type": "http",
  "tags": { "application": "site_public" },
  "schedule": { "interval": "15m" },
  "timeout": "10s",
  "http": {
    "method": "GET",
    "url": "http://www.example.com/",
    "followRedirects": false,
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 301 },
      { "kind": "header", "path": "Location", "operator": "matches", "value":
   "^https://www\\.example\\.com/" }
    ]
  }
}
```

Dans la valeur JSON de l'expression régulière, le point est échappé deux fois (`\\.`) : une fois pour l'expression régulière, une fois pour la syntaxe JSON.

10.6.11 Exemple : point de contrôle de santé d'une application

De nombreuses applications exposent un point de contrôle de santé renvoyant un document JSON. L'exemple suivant interroge le point de santé fictif d'une application interne, protégée par une AC d'entreprise et par un jeton d'accès stocké comme secret.

```json
{
  "id": "crm-sante",
  "name": "CRM : point de contrôle de santé",
  "type": "http",
  "tags": { "application": "crm", "environment": "production" },
  "schedule": { "interval": "1m" },
  "timeout": "10s",
  "secrets": ["CRM_HEALTH_TOKEN"],
  "http": {
    "method": "GET",
    "url": "https://crm.example.internal/actuator/health",
    "headers": {
      "Accept": "application/json",
      "Authorization": "Bearer ${secret:CRM_HEALTH_TOKEN}"
    },
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "header", "path": "Content-Type", "operator": "contains", "value": "application/json"
    },
      { "kind": "jsonPath", "path": "$.status", "operator": "eq", "value": "UP" },
      { "kind": "jsonPath", "path": "$.components.db.status", "operator": "eq", "value": "UP" }
    ],
    "tls": { "verify": true, "caRef": "ac-interne-example", "certificateExpiryWarningDays": 30 }
  }
}
```

Le secret `CRM_HEALTH_TOKEN` est déclaré dans la liste `secrets` du contrôle. Il n'est transmis au moteur d'exécution que pour ce contrôle, et sa valeur n'apparaît ni dans les résultats ni dans les journaux (chapitre 9).

> **Information** — Pour une authentification HTTP Basic, construisez l'en-tête `Authorization: Basic ${secret:NOM}`. Le secret contient alors la chaîne `utilisateur:motdepasse` encodée en Base64, et non le mot de passe en clair.

10.6.12 Exemple : téléchargement d'un fichier publié

La méthode `HEAD` vérifie la présence d'un fichier volumineux sans le télécharger. Ici, il s'agit d'un catalogue fictif publié chaque nuit.

```json
{
  "id": "catalogue-publie",
  "name": "Catalogue tarifaire : fichier publié",
  "type": "http",
  "tags": { "application": "catalogue" },
  "schedule": { "cron": "0 7-19 * * 1-5", "timezone": "Europe/Paris" },
  "timeout": "20s",
  "http": {
    "method": "HEAD",
    "url": "https://fichiers.example.com/catalogue/catalogue-courant.csv",
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "header", "path": "Content-Length", "operator": "exists" }
    ]
  }
}
```

10.6.13 Résultat attendu

```json
{
  "id": "01M7SH7FTQAA30DSWTVCMWPNPD",
  "checkId": "site-public-accueil",
  "checkRevision": 1,
  "status": "OK",
  "startedAt": "2025-11-19T09:12:00.087Z",
  "durationMs": 640,
  "measurements": {
    "http.status_code": 200,
    "http.dns_ms": 20,
    "http.connect_ms": 30,
    "http.tls_ms": 90,
    "http.ttfb_ms": 450,
    "http.transfer_ms": 50,
    "tls.certificate_days_remaining": 74
  }
}
```

En cas d'échec, le résultat précise l'assertion en cause. Exemple de message, consultable dans le détail de l'exécution (chapitre 17). Ce bloc est un **extrait (projection)** limité aux champs `status` et `error` ; la réponse complète est illustrée au chapitre 17, § 17.4 :

```json
{
  "status": "CRITICAL",
  "error": {
    "type": "assertion_failed",
    "message": "Assertion body contains « Bienvenue sur Example » : texte absent de la réponse (28
    413 octets reçus)"
  }
}
```

10.6.14 Erreurs courantes

Symptôme	Cause probable	Action
`CRITICAL` / `tls_error` sur un site interne	Certificat émis par une AC interne non importée sur la sonde	Importer l'AC dans la section `trust` (§ 10.6.8)
`CRITICAL` / `tls_error` après une intervention	Chaîne de certificats incomplète sur le serveur (certificat intermédiaire manquant). Les navigateurs la compensent parfois, la sonde non	Corriger la configuration du serveur web pour qu'il présente la chaîne complète
`CRITICAL` / `http_status` avec le code 403	Le pare-feu applicatif (WAF) ou un filtrage géographique bloque la sonde	Faire autoriser l'adresse de la sonde ; ajouter un `User-Agent` identifiable
`CRITICAL` / `http_status` avec le code 407	Un proxy exige une authentification	Configurer le proxy de la sonde (chapitre 7), ou l'exclure pour les destinations internes (liste d'exclusions, ou `http.proxy: "none"` dans le contrôle)
`CRITICAL` / `assertion_failed` sur `body` alors que la page s'affiche dans un navigateur	Contenu généré par JavaScript côté navigateur : absent de la réponse HTTP brute	Choisir un texte présent dans le HTML initial, ou utiliser un scénario navigateur (partie IV)
`CRITICAL` / `http_status` avec le code 429	Le serveur limite le nombre de requêtes	Réduire la fréquence du contrôle ; faire exempter la sonde des limites
Assertion `jsonPath` en échec alors que la valeur semble correcte	Type différent (nombre `200` contre texte `"200"`, booléen `true` contre texte `"true"`) ou chemin incorrect	Vérifier le type de la valeur dans la réponse brute affichée par le mode essai
États alternant entre `OK` et `WARNING`	`warningDuration` proche du temps de réponse habituel	Fixer le seuil à partir des percentiles observés (chapitre 17)
`CRITICAL` / `timeout` uniquement aux heures de pointe	Saturation de l'application ou du réseau	Analyser les phases ; c'est souvent un incident réel à signaler

10.7 Plugins Nagios (`nagios`)

10.7.1 Objectif et intérêt

Nagios est un logiciel de supervision historique. Son format de **plugins**, de petits programmes autonomes qui réalisent une vérification et renvoient un résultat normalisé, est devenu un standard de fait, repris par Centreon, Icinga, Naemon, Shinken et de nombreux autres outils. Des milliers de plugins existent.

Le contrôle `nagios` permet à la sonde d'exécuter ces plugins. Ce type répond à deux besoins :

- **couvrir des protocoles** que les contrôles natifs ne traitent pas en profondeur : dialogue SMTP complet avec STARTTLS, LDAP avec recherche, temps NTP, SSH, FTP, IMAP… ;
- **réutiliser l'existant** : une entreprise qui dispose de plugins personnalisés pour ses applications peut les exécuter depuis la sonde, au plus près des utilisateurs (édition Professional).

10.7.2 Contrat d'un plugin Nagios

Un plugin respecte deux conventions.

1. Le code de retour indique l'état :

Code de retour	État Nagios	État de la sonde
0	OK	`OK`
1	WARNING	`WARNING`
2	CRITICAL	`CRITICAL`
3	UNKNOWN	`UNKNOWN`
Autre valeur, ou arrêt anormal	—	`UNKNOWN` (`plugin_error`)

2. La sortie standard contient une ligne de texte lisible, éventuellement suivie, après le caractère `|`, de **données de performance** (*perfdata*) :

```
SMTP OK - 0.112 sec. response time|time=0.112094s;;;0.000000
```
TEXTE

Chaque donnée de performance suit le format :

```
'libellé'=valeur[unité];[seuil_warning];[seuil_critical];[min];[max]
```
TEXTE

Par exemple, `rta=12.400000ms;100.000000;500.000000;0.000000` indique un temps d'aller-retour moyen (`rta`) de 12,4 ms, un seuil d'avertissement à 100 ms, un seuil critique à 500 ms et une valeur minimale de 0.

La sonde conserve la sortie texte dans le résultat de l'exécution (4 096 caractères au maximum, secrets masqués). Elle n'exporte sous forme de métriques que les données de performance explicitement listées dans `perfdataExport` (§ 10.7.5).

10.7.3 Plugins fournis

L'image de la sonde intègre les plugins du projet **Monitoring Plugins** (paquet Debian `monitoring-plugins`), le jeu de plugins standard de l'écosystème Nagios. La liste exacte des plugins présents dans l'image est indiquée dans les notes de version. Les plus utiles depuis une sonde sont les suivants.

Plugin	Usage	Options principales
`check_http` / `check_curl`	Vérification HTTP avancée (contenu, redirections, certificat)	`-H` hôte, `-u` chemin, `-S` HTTPS, `--sni`, `-s` texte attendu, `-C` jours avant expiration du certificat
`check_smtp`	Dialogue SMTP, y compris STARTTLS	`-H` hôte, `-p` port, `-S` STARTTLS, `-e` réponse attendue
`check_imap`, `check_pop`	Disponibilité des services de messagerie en réception	`-H` hôte, `-p` port, `-S` SSL
`check_ldap`	Connexion et recherche dans un annuaire LDAP	`-H` hôte, `-b` base de recherche, `-T` STARTTLS, `-S` LDAPS
`check_ssh`	Disponibilité et version d'un service SSH	`-H` hôte, `-p` port, `-r` version attendue
`check_ntp_time`	Écart de l'heure d'un serveur NTP par rapport à l'horloge locale	`-H` serveur, `-w` / `-c` seuils d'écart en secondes
`check_dig`	Requête DNS détaillée avec vérification de la réponse	`-H` serveur DNS, `-l` nom, `-T` type, `-a` réponse attendue

Plugin	Usage	Options principales
`check_ftp`	Disponibilité d'un serveur FTP	`-H` hôte, `-p` port
`check_ping`	Ping avec seuils de latence et de perte. Il dépend de la commande `ping`, qui exige des privilèges réseau absents de l'environnement isolé : préférez le contrôle natif `icmp` (§ 10.3)	`-H` hôte, `-w` / `-c` sous la forme `latence,perte%`

Chaque plugin documente ses options avec l'argument `--help`. La documentation de référence est publiée par le projet Monitoring Plugins (monitoring-plugins.org).

> **Attention** — Certains plugins requièrent des privilèges élevés, car ils utilisent des sockets bruts ou des ports réservés : `check_icmp` et `check_dhcp`, par exemple. L'environnement d'exécution isolé de la sonde ne dispose pas de ces privilèges, qui ne lui sont pas accordés afin de maintenir son isolation. `check_icmp` et `check_dhcp` ne sont donc pas pris en charge. Pour le ping, utilisez le contrôle natif `icmp` (§ 10.3). Les plugins qui vérifient les ressources **locales** d'un serveur (`check_disk`, `check_load`, `check_procs`…) n'ont pas de sens sur la sonde, puisqu'ils mesureraient la sonde elle-même et non vos serveurs.

10.7.4 Paramètres

Paramètre	Obligatoire	Description
`nagios.plugin`	Oui	Nom de l'exécutable, **sans chemin** (lettres, chiffres, `_`, `.`, `-` ; 64 caractères au maximum). La sonde le recherche uniquement dans les répertoires de plugins autorisés
`nagios.args`	Non	Liste des arguments, **un élément par argument** (32 au maximum, 512 caractères chacun)
`nagios.perfdataExport`	Non	Libellés des données de performance à convertir en métriques (10 au maximum)

10.7.5 Exécution sécurisée

La sonde exécute les plugins **sans interpréteur de commandes** (*shell*) : le programme est lancé directement, avec la liste d'arguments fournie. Ce choix de conception protège la sonde contre les injections de commandes, mais il a des conséquences pratiques :

- chaque argument est un élément distinct de la liste `args` : `["-H", "smtp.example.internal"]` et non `["-H smtp.example.internal"]` ;
- les constructions propres au shell sont **sans effet** : redirections (`>`, `2>&1`), enchaînements (`;`, `&&`), tubes (`|`), variables (`$HOME`), substitutions (`$(…)`) ;
- un plugin ne peut pas être désigné par un chemin arbitraire.

Les plugins s'exécutent dans l'environnement isolé de la sonde, sous un compte système dédié, sans accès à la base de données de la sonde ni aux secrets non déclarés. Leur consommation de mémoire et de processeur est limitée.

Délais. Le timeout du contrôle (paramètre commun `timeout`) interrompt le plugin s'il ne s'est pas terminé. La plupart des plugins disposent aussi de leur propre délai, l'option `-t`. Fixez-le **inférieur** au timeout du contrôle : le plugin s'arrête alors proprement et renvoie un message explicite, au lieu d'être interrompu par la sonde.

Secrets. La référence `${secret:NOM}` est substituée dans les arguments `nagios.args` : elle permet de transmettre un mot de passe à un plugin. Le secret doit être déclaré dans la liste `secrets` du contrôle.

> **Avertissement** — Un argument de ligne de commande peut être visible par d'autres processus du système pendant l'exécution du plugin. Lorsqu'un plugin propose de lire ses identifiants depuis un fichier ou une variable d'environnement, préférez cette méthode. Dans tous les cas, utilisez des comptes de supervision aux droits minimaux.

Conversion des données de performance. Pour chaque libellé listé dans `perfdataExport`, la sonde publie la valeur sous la métrique `sva.plugin.perfdata`, avec l'attribut `sva.perfdata.label` (chapitre 18). Les libellés non listés restent visibles dans la sortie texte, mais ne produisent pas de métrique. Cette sélection explicite évite qu'un plugin produisant des libellés variables (un par fichier, par utilisateur…) ne multiplie le nombre de séries de métriques.

10.7.6 Exemple : messagerie sortante avec STARTTLS

```json
{
  "id": "smtp-soumission-starttls",
  "name": "Messagerie : soumission SMTP avec STARTTLS",
  "type": "nagios",
  "tags": { "application": "messagerie" },
  "schedule": { "interval": "5m" },
  "timeout": "30s",
  "nagios": {
    "plugin": "check_smtp",
    "args": ["-H", "smtp.example.internal", "-p", "587", "-S", "-w", "2", "-c", "5", "-t", "20"],
    "perfdataExport": ["time"]
  }
}
```

Le plugin se connecte au port 587, négocie le chiffrement par STARTTLS (`-S`), et renvoie `WARNING` au-delà de 2 secondes et `CRITICAL` au-delà de 5 secondes. Son propre délai (`-t 20`) est inférieur au timeout du contrôle (30 secondes).

10.7.7 Exemple : heure d'un serveur NTP interne

Une dérive de l'heure perturbe l'authentification Kerberos, la validité des jetons et la corrélation des journaux. Ce contrôle vérifie que le serveur NTP de l'entreprise délivre une heure cohérente avec celle de la sonde.

```json
{
  "id": "ntp-interne-ecart",
  "name": "Serveur NTP interne : écart de temps",
  "type": "nagios",
  "tags": { "domaine": "infrastructure" },
  "schedule": { "interval": "15m" },
  "timeout": "20s",
  "nagios": {
    "plugin": "check_ntp_time",
    "args": ["-H", "ntp1.example.internal", "-w", "0.5", "-c", "1", "-t", "10"],
    "perfdataExport": ["offset"]
  }
}
```

> **Information** — Ce contrôle compare l'heure du serveur NTP à celle de la sonde, elle-même synchronisée par NTP (chapitre 7). Si la sonde utilise ce même serveur comme unique source de temps, l'écart sera toujours faible. Pour un contrôle significatif, synchronisez la sonde sur une source indépendante, ou comparez plusieurs serveurs entre eux.

10.7.8 Exemple : recherche dans un annuaire LDAP

```json
{
  "id": "annuaire-recherche",
  "name": "Annuaire : connexion et recherche LDAP",
  "type": "nagios",
  "tags": { "application": "annuaire" },
  "schedule": { "interval": "5m" },
  "timeout": "30s",
  "nagios": {
    "plugin": "check_ldap",
    "args": ["-H", "ldap.example.internal", "-b", "dc=example,dc=internal", "-T", "-w", "1", "-c",
    "3", "-t", "20"],
    "perfdataExport": ["time"]
  }
}
```

10.7.9 Résultat attendu

```json
{
  "id": "01M7SHP4MTG5ZRSMPT8FDSNWZ5",
  "checkId": "smtp-soumission-starttls",
  "checkRevision": 1,
  "status": "OK",
  "startedAt": "2025-11-19T09:20:00.154Z",
  "durationMs": 214,
  "output": "SMTP OK - 0.112 sec. response time",
  "measurements": {
    "perfdata.time": 0.112094
  }
}
```

10.7.10 Plugins personnalisés (édition Professional)

À partir de l'édition Professional, un administrateur peut ajouter ses propres plugins. Le plugin se dépose par `POST /api/v1/plugins?name=<nom>` : le corps de la requête contient le fichier exécutable (`application/octet-stream`, 1 Mo au maximum). Le nom doit commencer par `check_` et ne contenir que des minuscules, des chiffres et des tirets bas (`check_[a-z0-9_]`, 40 caractères au plus après le préfixe). Un script doit commencer par sa ligne d'interpréteur (`#!/bin/sh` ou `#!/usr/bin/env python3`). La liste des plugins déposés s'obtient par `GET /api/v1/plugins`, et `DELETE /api/v1/plugins/{name}` retire un plugin ; ce retrait est refusé (`409`) tant qu'un contrôle l'utilise. Chaque dépôt et chaque retrait sont journalisés. Le dépôt et le retrait d'un plugin personnalisé sont réservés au rôle `admin`, car ils reviennent à installer ou à modifier du code exécutable sur le réseau de l'entreprise (chapitre 9) : un utilisateur `operator` reçoit une réponse `403`.

Un plugin personnalisé doit respecter les règles suivantes :

Règle	Justification
Respecter les codes de retour 0 à 3 et le format de sortie du § 10.7.2	Interprétation correcte de l'état
Être non interactif : aucune saisie au clavier, aucune confirmation	La sonde n'attend aucune interaction
Se terminer de lui-même avant le timeout du contrôle	Éviter les interruptions forcées (`CRITICAL`, `timeout`)
Ne dépendre que des interpréteurs et bibliothèques présents dans l'image (interpréteurs `sh`, fourni par `dash`, et `python3` ; utilitaire `curl`)	L'environnement isolé n'installe aucun paquet supplémentaire
N'écrire aucun fichier en dehors de son répertoire temporaire	Isolation et protection de la carte microSD

Règle	Justification
Ne jamais afficher de secret dans sa sortie	La sortie est stockée et peut être exportée
Produire des libellés de perfdata **fixes**	Maîtrise de la cardinalité des métriques

Squelette pédagogique d'un plugin en shell POSIX, qui vérifie l'âge d'un horodatage publié par une application sur une URL texte. Exemple fictif :

SHELL

```
#!/bin/sh
# check_fraicheur_export — exemple pédagogique de plugin personnalisé.
# Usage : check_fraicheur_export <URL> <seuil_warning_s> <seuil_critical_s>
# L'URL renvoie l'horodatage Unix (en secondes) du dernier export réussi.

URL="$1"; WARN="$2"; CRIT="$3"

if [ -z "$URL" ] || [ -z "$WARN" ] || [ -z "$CRIT" ]; then
  echo "UNKNOWN - arguments manquants"
  exit 3
fi

# Récupération de l'horodatage (délai maximal 10 s)
DERNIER=$(curl -fsS --max-time 10 "$URL") || {
  echo "CRITICAL - impossible de lire $URL"
  exit 2
}

case "$DERNIER" in
  ''|*[!0-9]*) echo "UNKNOWN - réponse inattendue"; exit 3 ;;
esac

AGE=$(( $(date +%s) - DERNIER ))
PERF="age=${AGE}s;${WARN};${CRIT};0"

if [ "$AGE" -ge "$CRIT" ]; then
  echo "CRITICAL - dernier export il y a ${AGE} s|${PERF}"; exit 2
elif [ "$AGE" -ge "$WARN" ]; then
  echo "WARNING - dernier export il y a ${AGE} s|${PERF}"; exit 1
fi
echo "OK - dernier export il y a ${AGE} s|${PERF}"
exit 0
```

Ce squelette utilise l'utilitaire `curl`, présent dans l'environnement d'exécution isolé. Le contrôle correspondant :

JSON

```
{
  "id": "export-comptable-fraicheur",
  "name": "Export comptable : fraîcheur du dernier export",
  "type": "nagios",
  "tags": { "application": "comptabilite" },
  "schedule": { "interval": "10m" },
  "timeout": "30s",
  "nagios": {
    "plugin": "check_fraicheur_export",
    "args": ["https://compta.example.internal/export/dernier", "7200", "14400"],
    "perfdataExport": ["age"]
  }
}
```

10.7.11 Erreurs courantes

Symptôme	Cause probable	Action
`UNKNOWN` / `plugin_error` dès la première exécution	Plugin introuvable dans les répertoires autorisés, nom mal orthographié, fichier non exécutable	Vérifier le nom exact ; pour un plugin personnalisé, vérifier son dépôt et ses droits d'exécution
`UNKNOWN` avec un message d'usage du plugin	Arguments regroupés dans un seul élément de `args`	Séparer chaque option et chaque valeur en éléments distincts
Le plugin fonctionne en ligne de commande sur un serveur, mais pas sur la sonde	Dépendance absente de l'image, privilèges requis, ou construction de shell utilisée (tube, redirection)	Vérifier les dépendances ; réécrire l'appel sans shell ; préférer un contrôle natif
`CRITICAL` / `timeout` après exactement la durée du timeout	Plugin bloqué, ou délai du plugin supérieur au timeout du contrôle	Fixer l'option `-t` du plugin en dessous du timeout du contrôle
Aucune métrique de perfdata dans la plateforme d'observabilité	Libellé absent de `perfdataExport`, ou orthographe différente (sensible à la casse)	Copier le libellé exact depuis la sortie du plugin, visible dans le résultat de l'exécution
Sortie tronquée	Sortie supérieure à 4 096 caractères	Réduire la verbosité du plugin ; seule la première ligne est utile à l'état

10.8 Synthèse

Besoin	Contrôle recommandé	Section
Joignabilité d'un équipement réseau	`icmp`	§ 10.3
Service non HTTP en écoute	`tcp`	§ 10.4
Bannière ou dialogue texte simple	`tcp` avec `send` / `expect`	§ 10.4.5
Service chiffré dès la connexion	`tcp` avec `tls`	§ 10.4.7
Service chiffré par STARTTLS	`nagios` (`check_smtp -S`, `check_ldap -T`…)	§ 10.7
Résolution d'un nom, valeur d'un enregistrement	`dns`	§ 10.5
Page web, point de santé, redirection	`http`	§ 10.6
Expiration des certificats	`http` avec `certificateExpiryWarningDays`	§ 10.6.7
Protocole spécifique couvert par un plugin standard	`nagios`	§ 10.7.3
Vérification propre à l'entreprise	`nagios`, plugin personnalisé (Professional)	§ 10.7.10
Réponse fonctionnelle d'une API, enchaînement de requêtes	`http`, `api`	Chapitre 11
Cohérence des données métier	`database`	Chapitre 11
Parcours utilisateur complet	`browser`	Partie IV

Chapitre 11 — Contrôles applicatifs et métier

Les contrôles techniques du chapitre 10 vérifient que les services répondent. Ce chapitre traite d'une question différente : les applications **rendent-elles le service attendu** ? Une API qui répond `200` avec un résultat erroné, une base de données accessible dont les traitements nocturnes n'ont pas tourné, une activité commerciale qui s'effondre sans qu'aucun serveur ne soit en panne : ces situations échappent à la supervision technique et sont précisément celles que les contrôles applicatifs et métier permettent de détecter.

Ce chapitre présente :

- la supervision d'API avec le contrôle `http` et avec le contrôle `api`, qui enchaîne plusieurs requêtes ;
- la supervision de web services SOAP ;
- la conception de contrôles fonctionnels robustes ;
- les contrôles de bases de données (`database`) ;
- la publication d'indicateurs métier et la détection d'anomalies.

Les parcours dans un navigateur, troisième niveau de la supervision applicative, font l'objet de la partie IV.

> **Information** — Les contrôles `api` et `database` requièrent l'édition **Professional** ou supérieure. Le contrôle `http` avec assertions de contenu est disponible dès l'édition Essential. Tous les exemples de ce chapitre sont fictifs : domaines en `example.com` et `example.internal`, comptes et données de test.

11.1 De la disponibilité technique à la réussite fonctionnelle

Une application peut être **techniquement disponible** sans être **fonctionnellement utilisable**. Le tableau suivant illustre l'écart entre ces deux niveaux et le type de contrôle qui permet de le détecter.

Situation	Contrôle technique	Contrôle applicatif ou métier	Contrôle adapté
L'API de stock répond `200`, mais tous les articles apparaissent indisponibles à la suite d'une erreur de synchronisation	`OK` : le serveur répond	`CRITICAL` : l'article témoin, toujours en stock, apparaît indisponible	`http` + `jsonPath`
L'authentification par jeton est cassée après une mise à jour du serveur d'identité	`OK` : les deux serveurs répondent	`CRITICAL` : l'obtention du jeton ou l'appel authentifié échoue	`api`
Le web service d'annuaire répond, mais ne trouve plus aucune adresse	`OK` : réponse SOAP valide	`CRITICAL` : l'adresse témoin n'est plus trouvée	`http` + `xpath`
Le traitement nocturne de facturation n'a pas tourné	`OK` : la base répond	`CRITICAL` : la dernière exécution réussie date de plus de 26 heures	`database`
Les commandes chutent de 70 % à cause d'un moyen de paiement défaillant chez un prestataire	`OK` : tous les serveurs fonctionnent	`CRITICAL` : le volume de commandes de l'heure est anormalement bas	`database` + indicateur métier
Des messages s'accumulent dans une file de traitement	`OK` : le serveur de files répond	`WARNING` à partir de 100 messages en attente depuis plus de 15 minutes, `CRITICAL` à partir de 500 (§ 11.6.8)	`database` ou `http` sur l'API d'administration de la file

La supervision applicative **complète** la supervision technique, elle ne la remplace pas. Lorsqu'un contrôle applicatif échoue, les contrôles techniques du même service indiquent si la cause est une panne d'infrastructure (chapitre 10, § 10.2.2) ou un dysfonctionnement purement fonctionnel.

11.2 Superviser une API REST avec le contrôle `http`

11.2.1 Principe

Une **API REST** (*Representational State Transfer*) est une interface de programmation accessible par HTTP, qui échange généralement des documents JSON. La plupart des applications modernes en exposent une, à destination de leur interface web, de leurs applications mobiles ou de partenaires.

Le contrôle `http`, décrit en détail au chapitre 10 (§ 10.6), suffit lorsqu'**une seule requête** permet de vérifier le bon fonctionnement : consultation d'une ressource témoin, recherche, calcul. Sa valeur applicative tient aux **assertions sur le contenu** : il ne vérifie pas seulement que l'API répond, mais qu'elle répond **juste**.

11.2.2 Choisir une donnée témoin

Un contrôle fonctionnel efficace repose sur une **donnée témoin** : un objet dont vous connaissez la valeur attendue, stable dans le temps et réservé à la supervision. Par exemple :

- un article témoin, toujours en stock, au prix fixe ;
- un client de test, avec une adresse connue ;
- un compte de démonstration, avec un solde figé ;
- une recherche dont le résultat est connu à l'avance.

> ✓ **Conseil** — Créez les données témoins en accord avec les équipes applicatives et documentez-les. Une donnée témoin supprimée ou modifiée lors d'un nettoyage provoque une fausse alerte. Une donnée témoin trop « réelle » peut fausser les statistiques métier : une commande de test comptabilisée dans le chiffre d'affaires, par exemple. Les applications prévoient souvent un marqueur pour exclure ces données des rapports.

11.2.3 Exemple : consultation d'une ressource témoin

L'API de stock fictive doit indiquer que l'article témoin `ARTICLE-TEMOIN` est disponible, avec une quantité positive et un prix inchangé.

JSON

```
{
  "id": "api-stock-article-temoin",
  "name": "API de stock : article témoin",
  "description": "Vérifie que l'API de stock renvoie l'état connu de l'article témoin (exemple
    fictif).",
  "type": "http",
  "tags": { "application": "stock", "environment": "production" },
  "schedule": { "interval": "1m" },
  "timeout": "10s",
  "retry": { "attempts": 1, "interval": "20s" },
  "thresholds": { "warningDuration": "800ms" },
  "secrets": ["API_STOCK_TOKEN"],
  "http": {
    "method": "GET",
    "url": "https://api.example.com/v1/stock/ARTICLE-TEMOIN",
    "headers": {
      "Accept": "application/json",
```

```
      "Authorization": "Bearer ${secret:API_STOCK_TOKEN}"
    },
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "header", "path": "Content-Type", "operator": "contains", "value": "application/json"
    },
      { "kind": "jsonPath", "path": "$.reference", "operator": "eq", "value": "ARTICLE-TEMOIN" },
      { "kind": "jsonPath", "path": "$.available", "operator": "eq", "value": true },
      { "kind": "jsonPath", "path": "$.quantity", "operator": "gt", "value": 0 },
      { "kind": "jsonPath", "path": "$.price.amount", "operator": "eq", "value": 19.9 },
      { "kind": "duration", "operator": "lt", "value": 2000 }
    ]
  }
}
```

Chaque assertion couvre un risque différent :

Assertion	Défaillance détectée
`status` égal à `200`	Erreur serveur, ressource introuvable, refus d'accès
`Content-Type` contient `application/json`	Page d'erreur HTML renvoyée par un proxy ou un répartiteur de charge à la place de l'API
`$.reference`	Réponse portant sur un autre objet : erreur de routage, cache corrompu
`$.available` et `$.quantity`	Données de stock incohérentes, synchronisation défaillante
`$.price.amount`	Erreur de tarification, chargement d'un mauvais référentiel de prix
`duration` inférieure à 2 000 ms	Dégradation sévère des performances

11.2.4 Exemple : recherche par requête POST

Certaines API de recherche attendent un corps de requête JSON. La requête suivante recherche un terme dont le résultat est connu. Elle ne crée et ne modifie aucune donnée.

JSON

```
{
  "id": "api-catalogue-recherche",
  "name": "API catalogue : recherche d'un terme témoin",
  "type": "http",
  "tags": { "application": "catalogue", "environment": "production" },
  "schedule": { "interval": "5m" },
  "timeout": "15s",
  "secrets": ["API_CATALOGUE_KEY"],
  "http": {
    "method": "POST",
    "url": "https://api.example.com/v2/catalogue/recherche",
    "headers": {
      "Content-Type": "application/json",
      "Accept": "application/json",
      "X-Api-Key": "${secret:API_CATALOGUE_KEY}"
    },
    "body": "{\"terme\": \"lampe de bureau témoin\", \"page\": 1, \"taille\": 10}",
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "jsonPath", "path": "$.total", "operator": "gte", "value": 1 },
      { "kind": "jsonPath", "path": "$.resultats[0].reference", "operator": "eq", "value": "LAMPE-
    TEMOIN-01" },
      { "kind": "body", "operator": "notContains", "value": "\"erreurs\"" }
    ]
  }
}
```

Le corps de la requête est une chaîne de caractères : les guillemets du document JSON envoyé sont échappés (`\"`).

11.2.5 Validation du contenu : choisir le bon opérateur

Besoin	Opérateur	Exemple
Valeur exacte connue et stable	`eq`	Référence, code de statut métier, booléen
Valeur numérique dans une plage	`gt`, `gte`, `lt`, `lte` (une assertion par borne)	Quantité positive, délai de livraison inférieur à 5 jours
Présence d'un champ, sans contrainte sur sa valeur	`exists`	Un identifiant de transaction est fourni
Format attendu, valeur variable	`matches`	Date au format `^\d{4}-\d{2}-\d{2}$`, identifiant de type `^CMD-[0-9]+$`
Absence d'un message d'erreur	`notContains` (sur `body`)	Pas de champ `"erreurs"`, pas de `"exception"`
Valeur différente d'un état anormal	`ne`	`$.mode` différent de `"maintenance"`

> ! **Attention** — N'écrivez pas d'assertion `eq` sur une valeur qui change légitimement : horodatage, identifiant généré, compteur, prix soumis à promotion. Le contrôle échouerait à chaque évolution normale. Utilisez `exists`, `matches` ou une plage numérique.

11.3 Enchaîner des requêtes avec le contrôle `api`

11.3.1 Principe

Certaines vérifications nécessitent **plusieurs requêtes successives**. La réponse de l'une fournit une donnée indispensable à la suivante. Cas typiques :

- obtenir un **jeton d'accès** auprès d'un serveur d'autorisation, puis appeler l'API protégée ;
- rechercher un objet, puis consulter son détail à partir de l'identifiant renvoyé ;
- vérifier la cohérence entre deux services : le stock affiché par le catalogue doit correspondre à celui de l'entrepôt.

Le contrôle `api` exécute une liste ordonnée de requêtes, de 1 à 20. Chaque requête :

- porte un **nom** (`name`), qui devient une **étape** du résultat : vous savez ainsi précisément quelle requête a échoué ;
- accepte tous les paramètres d'une requête `http` (§ 10.6.2), y compris ses propres assertions ;
- peut **extraire des variables** de sa réponse (`extract`), réutilisables dans les requêtes suivantes avec la syntaxe `${var:nom}`.

Le contournement du proxy global (`proxy: "none"`, § 10.6.2) se déclare pour l'ensemble du contrôle, dans `api.proxy`, et non requête par requête. De même, le paramètre commun `ipFamily` (chapitre 8, § 8.1.3) s'applique à toutes les requêtes du contrôle ; les cibles IPv6 suivent les mêmes règles que pour le contrôle `http` (chapitre 10, § 10.1.3).

Dès qu'une requête échoue (erreur réseau ou assertion non satisfaite), la suite est interrompue et le contrôle passe en `CRITICAL`. L'étape en échec et le type d'erreur sont indiqués dans le résultat.

11.3.2 Extraction de variables

Le paramètre `extract` associe un nom de variable à une expression, dont la nature est déclarée explicitement dans le champ `type` :

- `jsonpath` : expression **JSONPath**, pour une réponse JSON ;
- `xpath` : expression **XPath**, pour une réponse XML ;
- `regex` : **expression régulière** appliquée au corps de la réponse en texte. Si l'expression comporte un groupe de capture, la valeur extraite est celle du premier groupe ; sinon, c'est la totalité de la correspondance. Seule la première correspondance est retenue.

Champ	Obligatoire	Description	Défaut
`type`	Oui	`jsonpath`, `xpath` ou `regex`	—
`expression`	Oui	Expression d'extraction	—
`reveal`	Non	Si `true`, la valeur extraite apparaît en clair dans les résultats et les journaux	`false`

> **Attention** — Les deux graphies diffèrent par la casse : `"type": "jsonpath"` (tout en minuscules) dans `extract`, mais `"kind": "jsonPath"` (avec un P majuscule) dans les assertions. Une graphie inversée est refusée à l'enregistrement du contrôle.

```json
"extract": {
  "jeton": { "type": "jsonpath", "expression": "$.access_token" },
  "idCommande": { "type": "jsonpath", "expression": "$.commandes[0].id", "reveal": true }
}
```

Les variables extraites sont disponibles dans l'URL, les en-têtes et le corps des requêtes **suivantes** de la même exécution. Elles ne sont pas conservées d'une exécution à l'autre.

> **Information** — Une variable extraite peut contenir une donnée sensible, comme un jeton d'accès. La sonde masque dans les résultats et les journaux les valeurs des secrets déclarés, ainsi que, par défaut, celles des variables extraites. Réservez `"reveal": true` aux valeurs non sensibles utiles au diagnostic, comme un identifiant de commande témoin. Ne conservez dans les assertions et les messages que ce qui est nécessaire au diagnostic.

11.3.3 Exemple : jeton OAuth 2.0 puis appel d'une API protégée

OAuth 2.0 est le protocole d'autorisation le plus répandu pour les API. Dans le flux dit *client credentials*, une application s'authentifie auprès d'un serveur d'autorisation avec un identifiant et un secret client, puis obtient un **jeton d'accès** de courte durée. Ce jeton accompagne ensuite chaque appel à l'API.

Ce flux implique deux composants distincts : le serveur d'autorisation et l'API. Il peut échouer à chaque étape. Le contrôle `api` les vérifie ensemble, en séparant clairement les responsabilités.

```json
{
  "id": "api-commandes-oauth",
  "name": "API commandes : authentification OAuth 2.0 et consultation",
  "description": "Obtient un jeton (client credentials) puis consulte la commande témoin (exemple
    fictif).",
  "type": "api",
  "tags": { "application": "commandes", "environment": "production" },
  "schedule": { "interval": "5m" },
```

```
  "timeout": "30s",
  "retry": { "attempts": 1, "interval": "1m" },
  "thresholds": { "warningDuration": "3s" },
  "secrets": ["OAUTH_CLIENT_ID", "OAUTH_CLIENT_SECRET"],
  "api": {
    "requests": [
      {
        "name": "Obtenir un jeton d'accès",
        "method": "POST",
        "url": "https://auth.example.com/oauth2/token",
        "headers": { "Content-Type": "application/x-www-form-urlencoded" },
        "body":
    "grant_type=client_credentials&client_id=${secret:OAUTH_CLIENT_ID}&client_secret=${secret:OAUTH_C
    LIENT_SECRET}&scope=commandes.lecture",
        "assertions": [
          { "kind": "status", "operator": "eq", "value": 200 },
          { "kind": "jsonPath", "path": "$.access_token", "operator": "exists" },
          { "kind": "jsonPath", "path": "$.token_type", "operator": "eq", "value": "Bearer" }
        ],
        "extract": { "jeton": { "type": "jsonpath", "expression": "$.access_token" } }
      },
      {
        "name": "Rechercher la commande témoin",
        "method": "GET",
        "url": "https://api.example.com/v1/commandes?reference=CMD-TEMOIN-0001",
        "headers": {
          "Accept": "application/json",
          "Authorization": "Bearer ${var:jeton}"
        },
        "assertions": [
          { "kind": "status", "operator": "eq", "value": 200 },
          { "kind": "jsonPath", "path": "$.total", "operator": "eq", "value": 1 }
        ],
        "extract": { "idCommande": { "type": "jsonpath", "expression": "$.commandes[0].id", "reveal":
    true } }
      },
      {
        "name": "Consulter le détail de la commande",
        "method": "GET",
        "url": "https://api.example.com/v1/commandes/${var:idCommande}",
        "headers": {
          "Accept": "application/json",
          "Authorization": "Bearer ${var:jeton}"
        },
        "assertions": [
          { "kind": "status", "operator": "eq", "value": 200 },
          { "kind": "jsonPath", "path": "$.reference", "operator": "eq", "value": "CMD-TEMOIN-0001"
    },
          { "kind": "jsonPath", "path": "$.lignes[0].quantite", "operator": "eq", "value": 2 },
          { "kind": "duration", "operator": "lt", "value": 1500 }
        ]
      }
    ]
  }
}
```

Attention — Dans un corps `application/x-www-form-urlencoded`, les secrets sont substitués tels quels. Si une valeur contient des caractères réservés (`&`, `=`, `+`, `%`, espace), enregistrez le secret sous sa forme déjà encodée (par exemple `%2B` pour `+`), faute de quoi le serveur d'autorisation recevra une valeur tronquée ou altérée.

Lecture de l'exemple :

1. **Première requête.** Le jeton est demandé avec le périmètre (*scope*) minimal `commandes.lecture`. Le compte client utilisé pour la supervision ne doit disposer d'aucun droit d'écriture.
2. **Deuxième requête.** Le jeton extrait (`${var:jeton}`) authentifie la recherche. L'identifiant interne de la commande témoin est extrait de la réponse.
3. **Troisième requête.** Cet identifiant construit l'URL du détail, dont le contenu est vérifié.

Secrets et hôtes autorisés. Chaque secret ne peut être transmis qu'aux hôtes de sa liste `allowedHosts`, fixée par un administrateur (chapitre 9, § 9.8.1). Dans cet exemple, les secrets `OAUTH_CLIENT_ID` et `OAUTH_CLIENT_SECRET` doivent être créés avec `"allowedHosts": ["auth.example.com"]` : ils ne sont injectés que dans la requête adressée au serveur d'autorisation, jamais dans celles adressées à `api.example.com`. Ici, les secrets figurent dans le **corps** de la première requête : la liste s'applique à tout élément porteur d'un secret, et pas seulement aux en-têtes. Si une réponse redirige vers un autre hôte, les en-têtes porteurs d'un secret ne sont pas retransmis, et un corps porteur d'un secret renvoyé lors d'une redirection n'est transmis qu'à un hôte de la liste (chapitre 9, § 9.8.1). Un utilisateur `operator` peut référencer ces secrets dans un contrôle `api`, mais il ne peut ni les lire ni étendre la liste des hôtes autorisés.

> ! **Attention** — La variable `${var:jeton}`, extraite d'une réponse, n'est pas un secret de la sonde : la liste `allowedHosts` ne la concerne pas. Vérifiez que les requêtes qui l'emploient visent uniquement l'API prévue.

11.3.4 Résultat attendu et localisation de l'échec

Lorsque le serveur d'autorisation est défaillant, le résultat indique sans ambiguïté que le problème se situe à la première étape, et non dans l'API de commandes :

```json
{
  "checkId": "api-commandes-oauth",
  "status": "CRITICAL",
  "durationMs": 10042,
  "error": {
    "type": "timeout",
    "step": "Obtenir un jeton d'accès",
    "message": "Délai dépassé lors de la requête POST https://auth.example.com/oauth2/token"
  },
  "steps": [
    { "index": 0, "name": "Obtenir un jeton d'accès", "status": "CRITICAL", "durationMs": 10042 }
  ]
}
```

Les durées et les échecs par étape sont publiés dans les métriques `sva.step.duration` et `sva.step.failures` (chapitre 18). Vous pouvez ainsi suivre séparément le temps de réponse du serveur d'autorisation et celui de l'API.

> ✓ **Conseil** — Donnez aux requêtes des **noms fixes et explicites**, rédigés du point de vue fonctionnel (« Obtenir un jeton d'accès »). Ne faites jamais figurer dans un nom une valeur variable, comme un identifiant ou une date : chaque nom distinct crée de nouvelles séries de métriques.

11.3.5 Erreurs courantes des contrôles `api`

Symptôme	Cause probable	Action
Échec à l'étape d'authentification avec le code 401	Secret client expiré, révoqué ou modifié	Renouveler le secret avec l'équipe responsable du serveur d'autorisation, puis le remplacer sur la sonde (chapitre 9)
Échec à l'étape d'authentification avec le code 400	Paramètres du flux incorrects (périmètre inconnu, type de flux non autorisé pour ce client)	Comparer le corps de la requête avec la documentation du serveur d'autorisation
Échec à la deuxième étape avec le code 401, alors que la première réussit	Variable mal référencée (`${var:…}` mal orthographié), ou jeton émis pour une autre audience	Vérifier le nom de la variable dans `extract` et dans l'en-tête ; vérifier l'audience du jeton
Échec avec le code 403	Le compte de supervision n'a pas les droits sur la ressource témoin	Ajuster les droits du compte, en restant en lecture seule
Échec d'authentification alors que le secret est correct	L'hôte de la requête ne figure pas dans la liste `allowedHosts` du secret : le secret n'a pas été injecté	Faire vérifier ou compléter la liste par un administrateur (chapitre 9, § 9.8)
Variable vide, URL invalide	Expression d'extraction ne correspondant pas à la réponse (structure modifiée, tableau vide)	Exécuter le contrôle en mode essai et examiner la réponse de l'étape précédente

11.4 Superviser un web service SOAP

11.4.1 Principe

SOAP (*Simple Object Access Protocol*) est un protocole d'échange de messages XML, encore très présent dans les systèmes d'information : progiciels, échanges interentreprises, services administratifs. Un appel SOAP est une requête HTTP `POST` dont le corps est une **enveloppe XML**, et dont la réponse est elle aussi une enveloppe XML.

La sonde supervise les web services SOAP avec le contrôle `http` :

- le corps de la requête contient l'enveloppe ;
- les en-têtes précisent le type de contenu et, selon la version de SOAP, l'action appelée ;
- les assertions `body` et `xpath` vérifient la réponse.

Le contrôle `api` permet d'enchaîner plusieurs appels SOAP.

11.4.2 Exemple : recherche d'une adresse dans un annuaire

Cet exemple reprend l'intention d'un scénario historique de supervision : vérifier qu'un web service d'annuaire de messagerie retrouve une adresse connue, ce qui prouve que le service fonctionne de bout en bout, jusqu'à sa source de données. Toutes les valeurs sont fictives :

- service `https://annuaire.example.internal/ws/annuaire` ;
- opération `rechercherAdresse` ;
- adresse témoin `test-supervision@example.com`.

```json
{
  "id": "ws-annuaire-recherche",
  "name": "Web service annuaire : recherche de l'adresse témoin",
  "description": "Appel SOAP 1.1 de l'opération rechercherAdresse sur l'adresse de test (exemple
    fictif).",
```

```
  "type": "http",
  "tags": { "application": "annuaire_messagerie", "environment": "production" },
  "schedule": { "interval": "5m" },
  "timeout": "15s",
  "retry": { "attempts": 1, "interval": "30s" },
  "http": {
    "method": "POST",
    "url": "https://annuaire.example.internal/ws/annuaire",
    "headers": {
      "Content-Type": "text/xml; charset=utf-8",
      "SOAPAction": "\"urn:annuaire#rechercherAdresse\""
    },
    "body": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><soapenv:Envelope
    xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\" xmlns:ann=\"urn:annuaire\">
    <soapenv:Body><ann:rechercherAdresse><ann:email>test-supervision@example.com</ann:email>
    </ann:rechercherAdresse></soapenv:Body></soapenv:Envelope>",
    "assertions": [
      { "kind": "status", "operator": "eq", "value": 200 },
      { "kind": "body", "operator": "notContains", "value": "Fault>" },
      { "kind": "xpath", "path": "//*[local-name()='trouve']", "operator": "eq", "value": "0" },
      { "kind": "xpath", "path": "//*[local-name()='email']", "operator": "eq", "value": "test-
    supervision@example.com" }
    ],
    "tls": { "verify": true, "caRef": "ac-interne-example" }
  }
}
```

Points d'attention :

Élément	Explication
`Content-Type: text/xml` et `SOAPAction`	En-têtes de SOAP 1.1. En SOAP 1.2, le type est `application/soap+xml` et l'action figure dans un paramètre `action` de ce même en-tête
`notContains` `Fault>`	Une erreur SOAP (*SOAP Fault*) peut être renvoyée avec le code HTTP 500, mais certains serveurs la renvoient avec le code 200. Cette assertion couvre les deux cas
`//*[local-name()='trouve']`	Les réponses SOAP utilisent des **espaces de noms** XML dont le préfixe varie d'un serveur à l'autre. La fonction `local-name()` désigne un élément par son nom local, quel que soit son préfixe. C'est la forme la plus robuste
Corps sur une ligne	Le corps est une chaîne JSON : les guillemets de l'enveloppe sont échappés (`\"`). Préparez et testez l'enveloppe dans un outil dédié, puis convertissez-la en chaîne JSON

11.4.3 Migration des projets SoapUI

Les clients de la génération précédente de la solution utilisaient des projets SoapUI pour superviser leurs web services. Ces projets ne sont pas exécutables par la sonde. Leurs éléments se transposent comme suit :

Élément SoapUI	Équivalent sur la sonde
Projet et suite de tests	Un ou plusieurs contrôles `http` ou `api`
Étape de requête SOAP ou REST (*Test Step*)	Requête d'un contrôle `http`, ou requête nommée d'un contrôle `api`
Enveloppe SOAP de la requête	`http.body`
Assertion *Contains* / *Not Contains*	Assertion `body` avec `contains` / `notContains`
Assertion *XPath Match*	Assertion `xpath`
Assertion *Valid HTTP Status Codes*	Assertion `status`
Assertion *Response SLA*	Assertion `duration`, ou `thresholds`

Élément SoapUI	Équivalent sur la sonde
Transfert de propriété entre étapes (*Property Transfer*)	`extract` et `${var:…}` du contrôle `api`
Script Groovy	Pas d'équivalent direct : plugin personnalisé (chapitre 10, § 10.7.10) ou découpage en contrôles plus simples

> **Information** — La transposition est manuelle. Elle est l'occasion de revoir les assertions : un simple *Contains* sur un fragment XML est souvent remplaçable par une assertion `xpath`, plus précise.

> **Évolution prévue** — Un outil d'import des projets SoapUI est prévu dans une version ultérieure pour accompagner la migration.

11.5 Concevoir des contrôles fonctionnels robustes

Un contrôle fonctionnel n'a de valeur que s'il est **fiable** : il doit échouer quand le service est réellement dégradé, et seulement dans ce cas. Les principes suivants s'appliquent aux contrôles `http`, `api` et `database`, ainsi qu'aux scénarios navigateur.

Principe	Mise en œuvre
Vérifier le résultat métier, pas seulement la réponse	Une assertion sur une valeur fonctionnelle (quantité, référence, état) en plus du code HTTP
Utiliser des données témoins dédiées	Objets réservés à la supervision, documentés, exclus des statistiques
Préférer des invariants à des valeurs exactes	« La quantité est positive » plutôt que « la quantité vaut 148 »
Rester en lecture seule	Aucune création, modification ou suppression de données en production (§ 11.9)
Couvrir les erreurs « silencieuses »	Assertions `notContains` sur les marqueurs d'erreur renvoyés avec un code 200
Séparer les responsabilités	Une étape nommée par composant (authentification, recherche, détail)
Calibrer les seuils de performance sur l'historique	Valeurs fondées sur les percentiles observés (chapitre 17)
Maintenir les contrôles avec les applications	Revue des contrôles à chaque évolution de l'API ou du schéma de données

> **Conseil** — Associez chaque contrôle fonctionnel à une **fiche descriptive** : ce qu'il vérifie, la donnée témoin utilisée, le responsable applicatif à prévenir, la conduite à tenir en cas d'échec. Le champ `description` du contrôle peut en contenir un résumé.

11.6 Contrôles de bases de données (`database`)

11.6.1 Principe et intérêt

Le contrôle `database` se connecte à une base de données, exécute une **requête SQL** ou une **procédure stockée**, puis vérifie le résultat au moyen de **règles**. Il répond à des questions que ni le réseau ni les API ne permettent de traiter :

- les traitements planifiés (*batchs*) ont-ils tourné, et ont-ils réussi ?
- des données s'accumulent-elles anormalement : messages en attente, commandes bloquées, rejets ?

- un volume d'activité est-il cohérent avec l'heure et le jour ?
- un référentiel est-il complet, et les données sont-elles cohérentes entre deux tables ?

Le contrôle `database` permet aussi de **publier des valeurs** issues de la requête sous forme d'indicateurs métier (§ 11.7).

11.6.2 Bases de données prises en charge

Le moteur principal de la sonde se connecte aux bases de données au moyen de pilotes **JDBC** (*Java Database Connectivity*), l'interface standard d'accès aux bases de données de la plateforme Java.

Moteur (`engine`)	SGBD	Port par défaut	Remarques
`postgresql`	PostgreSQL	5432	—
`mysql`	MySQL	3306	—
`mariadb`	MariaDB	3306	—
`sqlserver`	Microsoft SQL Server	1433	Chiffrement de la connexion : `tls`. Autres options de connexion, dont les instances nommées : propriétés JDBC `properties`, liste fermée
`oracle`	Oracle Database	1521	Le champ `database` désigne le nom de service Oracle

Les versions de chaque SGBD prises en charge dépendent des pilotes intégrés ; elles sont indiquées dans les notes de version.

11.6.3 Paramètres

Paramètre	Obligatoire	Description
`database.engine`	Oui	`postgresql`, `mysql`, `mariadb`, `sqlserver` ou `oracle`
`database.host`	Oui	Nom d'hôte ou adresse IP du serveur. Une adresse IPv6 est admise ; la famille d'adresses suit le paramètre commun `ipFamily` (chapitre 8, § 8.1.3, et chapitre 10, § 10.1.3)
`database.port`	Non	Port, si différent du port par défaut
`database.database`	Non	Nom de la base (ou nom de service Oracle)
`database.username`	Oui	Compte de connexion, en lecture seule (§ 11.6.4)
`database.passwordSecret`	Oui	Nom du secret contenant le mot de passe. Ce secret doit aussi figurer dans la liste `secrets` du contrôle, et l'hôte `database.host` (avec son port, le cas échéant) dans la liste `allowedHosts` du secret (chapitre 9, § 9.8.1)
`database.tls`	Non	Si `true`, chiffre la connexion au serveur (TLS). Défaut : `false`. Le chiffrement ne protège contre l'interception que si le certificat du serveur est vérifié (tableau ci-dessous)
`database.properties`	Non	Propriétés de connexion JDBC complémentaires (nom et valeur texte). Seules les clés d'une liste fermée, propre à chaque moteur, sont acceptées ; cette liste est indiquée dans les notes de version
`database.query`	L'un des deux	Requête SQL à exécuter (10 000 caractères maximum)
`database.procedure`	L'un des deux	Nom de la procédure stockée à appeler

Paramètre	Obligatoire	Description
`database.params`	Non	Paramètres de la procédure (nom et valeur)
`database.rules`	Oui	Règles appliquées au résultat (1 à 20)
`database.onEmpty`	Non	État produit lorsque le résultat ne contient aucune ligne : `OK`, `WARNING`, `CRITICAL` ou `UNKNOWN` (§ 11.6.6). Défaut : `UNKNOWN`
`database.exportValues`	Non	Colonnes publiées comme indicateurs métier (10 au maximum, § 11.7)

Le délai maximal (connexion et exécution) est fixé par le paramètre commun `timeout` du contrôle.

Sécurité de la connexion, délais et annulation selon le moteur. La sonde applique un délai de requête **côté pilote**, par la méthode JDBC standard `Statement.setQueryTimeout`. Ce délai seul ne suffit pas : si la connexion est perdue, le serveur peut poursuivre l'exécution. Fixez donc aussi une limite **côté serveur** sur le compte de supervision. Le tableau suivant rappelle, pour information, les mécanismes standards de chaque moteur et de son pilote JDBC de référence. Les options effectivement appliquées par la sonde lorsque `tls` vaut `true`, la correspondance entre `timeout` et le délai du pilote, et la liste des propriétés `properties` autorisées sont indiquées dans les notes de version.

Moteur	Vérification du certificat (option du pilote de référence)	Limite côté serveur	Annulation sur délai dépassé
PostgreSQL	`sslmode=verify-full` (pilote pgJDBC) : chaîne de confiance et nom d'hôte	`statement_timeout`, fixable pour le rôle (`ALTER ROLE … SET statement_timeout`, § 11.6.4)	Le pilote envoie une demande d'annulation au serveur par une connexion distincte
MySQL	`sslMode=VERIFY_IDENTITY` (MySQL Connector/J 8 et ultérieur)	`max_execution_time` (en millisecondes), variable de session ou globale, ou indication `/*+ MAX_EXECUTION_TIME(90000) */` dans la requête ; ne s'applique qu'aux instructions `SELECT`	Le pilote demande l'arrêt de la requête (`KILL QUERY`) par une connexion distincte
MariaDB	`sslMode=verify-full` (MariaDB Connector/J 3)	`max_statement_time` (en secondes), par session, globalement ou par compte (`ALTER USER … WITH MAX_STATEMENT_TIME 90`)	Mécanisme propre au pilote
SQL Server	`encrypt=true` et `trustServerCertificate=false` (pilote Microsoft JDBC) ; `hostNameInCertificate` si le nom du certificat diffère de l'hôte	Pas de délai d'exécution natif par compte équivalent à `statement_timeout` : la limite repose sur le délai du pilote	Le pilote envoie un signal d'interruption (*attention*) au serveur
Oracle Database	Connexion `TCPS` avec vérification du nom du serveur (`oracle.net.ssl_server_dn_match=true`)	Profil limitant le temps processeur par appel (`CPU_PER_CALL`, pris en compte si `RESOURCE_LIMIT` est actif), ou Resource Manager (directive `CANCEL_SQL`)	Le pilote demande l'annulation ; la requête se termine par l'erreur `ORA-01013`

Une annulation interrompt une requête ou une procédure **en cours de route**. Pour une requête de lecture qualifiée sans effet, y compris les fonctions qu'elle appelle (§ 11.6.5), l'interruption ne laisse aucune écriture métier. Elle n'est pas pour autant neutre : si le pilote n'obtient pas l'annulation (connexion perdue, demande non traitée par le

serveur), la requête peut se poursuivre côté serveur et conserver des ressources (processeur, mémoire, verrous, transaction ouverte) jusqu'à la limite côté serveur ou la fin de la session. Après une annulation ou un délai dépassé, la connexion doit être ramenée à un état sain (transaction annulée) ou fermée, et ses ressources libérées, avant toute réutilisation. Pour une procédure qui écrirait, des effets partiels pourraient subsister : c'est une raison supplémentaire d'exiger des procédures non modifiantes (§ 11.6.5).

> **Attention** — Faites valider la limite côté serveur par l'administrateur de la base : elle doit rester **supérieure** au `timeout` du contrôle, pour que l'erreur constatée soit celle de la sonde (`timeout`) et non une erreur SQL.

11.6.4 Préparer un compte de supervision en lecture seule

La sonde doit se connecter avec un **compte dédié**, disposant uniquement du droit de lecture sur les objets nécessaires. Ce compte limite les conséquences d'une erreur dans une requête ou d'une compromission des identifiants. Les commandes suivantes sont des exemples, à adapter et à faire exécuter par l'administrateur de la base. Les noms d'objets sont fictifs.

> **Avertissement** — Le compte de supervision est **dédié** à la sonde et ses droits sont **vérifiés côté serveur**, et non seulement déclarés : la sonde ne peut pas contrôler les droits du compte qu'on lui confie. Vérifiez aussi qu'une tentative d'écriture avec ce compte est **refusée**.

Faites vérifier par l'administrateur de la base les **droits effectifs** du compte, y compris ceux hérités de rôles, ceux accordés à tous les comptes (`PUBLIC`) et les droits par défaut sur les objets futurs. Outils usuels : `has_table_privilege` et la commande `\dp` de `psql` (PostgreSQL), `SHOW GRANTS FOR` (MySQL, MariaDB), `sys.fn_my_permissions` exécutée sous l'identité du compte (SQL Server), vues `SESSION_PRIVS` et `SESSION_ROLES` (Oracle). Renouvelez cette vérification à chaque évolution des rôles de la base.

PostgreSQL :

```sql
CREATE ROLE supervision_ro LOGIN PASSWORD '********';
GRANT CONNECT ON DATABASE ventes TO supervision_ro;
GRANT USAGE ON SCHEMA public TO supervision_ro;
GRANT SELECT ON public.commandes TO supervision_ro;
-- Limite la durée de toute requête de ce compte à 90 secondes : valeur supérieure
-- au timeout des contrôles qui utilisent ce compte (30 secondes au plus ici)
ALTER ROLE supervision_ro SET statement_timeout = '90s';
```

MySQL / MariaDB :

```sql
CREATE USER 'supervision_ro'@'192.0.2.50' IDENTIFIED BY '********';
GRANT SELECT ON ventes.commandes TO 'supervision_ro'@'192.0.2.50';
```

Le compte MySQL est restreint à l'adresse de la sonde (`192.0.2.50` dans cet exemple).

Microsoft SQL Server :

```sql
CREATE LOGIN supervision_ro WITH PASSWORD = '********';
USE Exploitation;
CREATE USER supervision_ro FOR LOGIN supervision_ro;
GRANT SELECT ON dbo.HistoriqueTraitements TO supervision_ro;
```

Oracle Database :

```sql
CREATE USER supervision_ro IDENTIFIED BY "********";
GRANT CREATE SESSION TO supervision_ro;
GRANT SELECT ON FACTURATION.SUIVI_LOTS TO supervision_ro;
GRANT EXECUTE ON FACTURATION.PKG_SUPERVISION TO supervision_ro;
```

> ! **Avertissement** — N'utilisez jamais un compte applicatif ou un compte d'administration de la base pour la supervision. Si la requête doit appeler une procédure stockée, vérifiez avec l'équipe responsable que cette procédure ne réalise aucune écriture.

11.6.5 Requête ou procédure stockée

Forme	Usage	Paramètre
Requête	Cas général : comptage, recherche de la dernière exécution, calcul d'écart	`query`
Procédure stockée	Logique de contrôle déjà écrite et maintenue par l'équipe applicative, ou accès aux tables non autorisé directement	`procedure` + `params`

Les deux formes sont exclusives : un contrôle utilise soit `query`, soit `procedure`.

Pour une procédure, les paramètres sont transmis par leur nom, avec des valeurs de type texte, nombre, booléen ou nulle :

```json
"procedure": "PKG_SUPERVISION.CONTROLE_LOTS",
"params": { "p_application": "FACTURATION", "p_fenetre_heures": 26 }
```

> ! **Avertissement** — Le droit d'exécuter une procédure peut donner accès à **davantage** que les droits du compte de supervision : un compte « en lecture seule » doté du droit `EXECUTE` peut écrire au travers d'une procédure.

Selon le moteur, une procédure s'exécute avec les droits de son propriétaire : c'est le comportement par défaut d'Oracle (`AUTHID DEFINER`) et de MySQL et MariaDB (`SQL SECURITY DEFINER`) ; c'est aussi le cas des fonctions PostgreSQL déclarées `SECURITY DEFINER` ; sous SQL Server, le chaînage des propriétaires (*ownership chaining*) permet à la procédure d'accéder aux objets du même propriétaire sans droit direct du compte.

Chaque procédure utilisée par la sonde doit être **qualifiée comme non modifiante** par l'équipe qui la maintient, avant sa mise en supervision et à chacune de ses évolutions :

- aucune écriture (`INSERT`, `UPDATE`, `DELETE`, `MERGE`), aucune modification de structure, aucune validation de transaction (`COMMIT`), y compris dans une table de journalisation ou d'audit ;
- aucune transaction autonome (par exemple `PRAGMA AUTONOMOUS_TRANSACTION` sous Oracle) ni effet de bord externe : envoi de courrier électronique, dépôt de message dans une file, appel d'un service, lancement d'un traitement ;
- même vérification pour les procédures et fonctions **appelées par** la procédure (droits transitifs), ainsi que pour toute fonction définie dans la base et appelée par une **requête de lecture** (`SELECT`) du contrôle : une telle fonction peut écrire, s'exécuter avec les droits de son propriétaire ou produire un effet externe, et se qualifie donc comme une procédure, moteur par moteur ;
- exécution compatible avec une interruption à tout instant (annulation sur délai dépassé, § 11.6.3).

La procédure doit renvoyer un **jeu de résultats** dont les colonnes sont nommées, puisque les règles s'appliquent à ces colonnes. Le mode de restitution du résultat selon le SGBD (curseur de sortie Oracle, jeu de résultats SQL Server…) est propre à chaque SGBD ; les conventions applicables sont indiquées dans les notes de version.

11.6.6 Règles

Une **règle** compare la valeur d'une colonne du résultat à une valeur de référence.

Élément	Valeurs	Description
`column`	Nom de colonne	Colonne du résultat à vérifier. Donnez un alias explicite dans la requête (`AS nb_commandes`)
`operator`	`eq`, `ne`, `lt`, `lte`, `gt`, `gte`	Égal, différent, strictement inférieur, inférieur ou égal, strictement supérieur, supérieur ou égal. Les comparaisons d'ordre ne s'appliquent qu'aux valeurs numériques
`value`	Nombre, texte ou booléen	Valeur de référence
`severity`	`critical` (défaut), `warning`	État produit si la règle n'est pas respectée
`scope`	`allRows` (défaut), `anyRow`, `firstRow`	Lignes concernées : toutes les lignes doivent respecter la règle, au moins une ligne, ou seulement la première

Calcul de l'état :

1. si au moins une règle de sévérité `critical` n'est pas respectée, l'état est `CRITICAL` (`assertion_failed`) ;
2. sinon, si au moins une règle de sévérité `warning` n'est pas respectée, l'état est `WARNING` ;
3. sinon, l'état est `OK`, sous réserve des seuils de durée (`thresholds`).

Une règle s'exprime comme la **condition normale** : elle décrit ce qui doit être vrai pour que tout aille bien. Exemple : « le nombre de commandes doit être supérieur à 0 » s'écrit `{ "column": "nb", "operator": "gt", "value": 0 }`.

Pour définir deux niveaux d'alerte sur une même colonne, déclarez deux règles : une règle `warning` au seuil le plus exigeant, une règle `critical` au seuil le plus large.

> **Attention** — Une requête qui ne renvoie **aucune ligne** ne permet pas d'évaluer les règles. L'état produit est alors fixé par le paramètre `onEmpty` : `UNKNOWN` par défaut, avec le type d'erreur `no_data`. Il est fortement recommandé d'écrire des requêtes qui renvoient **toujours exactement une ligne**, au moyen de fonctions d'agrégation (`COUNT`, `MAX`, `MIN`, `SUM`, `COALESCE`). Par exemple, `SELECT COUNT(*) AS nb …` renvoie `0` et non un résultat vide.

Erreurs d'exécution :

Situation	État	Type d'erreur
Serveur injoignable, base inexistante, erreur SQL	`CRITICAL`	`database_error`
Identifiants refusés	`CRITICAL`	`auth_failed`
Délai dépassé (connexion ou requête)	`CRITICAL`	`timeout`
Aucune ligne renvoyée	Selon `onEmpty` (`UNKNOWN` par défaut)	`no_data`
Règle non respectée	Selon `severity`	`assertion_failed`

11.6.7 Exemple : traitement nocturne (SQL Server)

L'application fictive d'exploitation enregistre chaque exécution de ses traitements dans une table d'historique. Le contrôle vérifie que le traitement de facturation a réussi au cours des dernières 26 heures. La marge de 2 heures au-delà d'un cycle quotidien absorbe les variations normales de l'heure de fin. Le contrôle vérifie aussi qu'aucun traitement n'est en erreur.

```json
{
  "id": "batch-facturation-nocturne",
  "name": "Traitement de facturation : dernière exécution réussie",
  "type": "database",
  "tags": { "application": "facturation", "environment": "production" },
  "schedule": { "cron": "*/30 6-20 * * *", "timezone": "Europe/Paris" },
  "timeout": "30s",
  "secrets": ["DB_EXPLOITATION_PASSWORD"],
  "database": {
    "engine": "sqlserver",
    "host": "sql-exploitation.example.internal",
    "port": 1433,
    "database": "Exploitation",
    "username": "supervision_ro",
    "passwordSecret": "DB_EXPLOITATION_PASSWORD",
    "query": "SELECT COALESCE(DATEDIFF(MINUTE, MAX(CASE WHEN Statut = 'SUCCES' THEN DateFin END),
    SYSUTCDATETIME()), 99999) AS minutes_depuis_succes, COALESCE(SUM(CASE WHEN Statut = 'ERREUR' AND
    DateFin > DATEADD(HOUR, -26, SYSUTCDATETIME()) THEN 1 ELSE 0 END), 0) AS erreurs_26h FROM
    dbo.HistoriqueTraitements WHERE Traitement = 'FACTURATION'",
    "rules": [
      { "column": "minutes_depuis_succes", "operator": "lte", "value": 1560, "severity": "critical",
    "scope": "firstRow" },
      { "column": "erreurs_26h", "operator": "eq", "value": 0, "severity": "warning", "scope":
    "firstRow" }
    ],
    "exportValues": [{ "column": "minutes_depuis_succes", "indicator":
    "facturation_minutes_depuis_succes" }]
  }
}
```

Décryptage :

- `COALESCE(…, 99999)` garantit une valeur même si aucun succès n'a jamais été enregistré. La règle échoue alors clairement.
- La règle `critical` porte sur l'absence de succès depuis plus de 1 560 minutes (26 heures).
- La règle `warning` signale un échec intermédiaire suivi d'une relance réussie : la situation est rétablie, mais mérite d'être examinée.
- La planification par expression `cron` limite le contrôle aux heures où une intervention est possible (de 6 h à 20 h). La planification est détaillée au chapitre 12.

11.6.8 Exemple : messages en attente (MariaDB)

```json
{
  "id": "file-integration-en-attente",
  "name": "File d'intégration : messages en attente",
  "type": "database",
  "tags": { "application": "integration" },
  "schedule": { "interval": "5m" },
  "timeout": "20s",
  "secrets": ["DB_INTEGRATION_PASSWORD"],
  "database": {
    "engine": "mariadb",
    "host": "db-integration.example.internal",
    "database": "integration",
    "username": "supervision_ro",
    "passwordSecret": "DB_INTEGRATION_PASSWORD",
    "query": "SELECT COUNT(*) AS en_attente_15min FROM messages WHERE etat = 'EN_ATTENTE' AND
   date_reception < NOW() - INTERVAL 15 MINUTE",
    "rules": [
      { "column": "en_attente_15min", "operator": "lt", "value": 100, "severity": "warning" },
      { "column": "en_attente_15min", "operator": "lt", "value": 500, "severity": "critical" }
    ],
    "exportValues": [{ "column": "en_attente_15min", "indicator": "messages_en_attente" }]
  }
}
```

À 120 messages bloqués depuis plus de 15 minutes, seule la règle `warning` est enfreinte : l'état est `WARNING`. À 600 messages, la règle `critical` l'est aussi : l'état est `CRITICAL`.

11.6.9 Exemple : vérification par une procédure stockée (Oracle)

```json
{
  "id": "facturation-controle-lots",
  "name": "Facturation : contrôle de cohérence des lots",
  "type": "database",
  "tags": { "application": "facturation" },
  "schedule": { "cron": "0 7,13 * * 1-5", "timezone": "Europe/Paris" },
  "timeout": "1m",
  "secrets": ["DB_FACTURATION_PASSWORD"],
  "database": {
    "engine": "oracle",
    "host": "ora-facturation.example.internal",
    "port": 1521,
    "database": "FACTPRD",
    "username": "supervision_ro",
    "passwordSecret": "DB_FACTURATION_PASSWORD",
    "procedure": "PKG_SUPERVISION.CONTROLE_LOTS",
    "params": { "p_application": "FACTURATION", "p_fenetre_heures": 26 },
    "rules": [
      { "column": "STATUT", "operator": "eq", "value": "OK", "scope": "allRows" }
    ]
  }
}
```

La procédure renvoie une ligne par lot contrôlé, avec une colonne `STATUT`. La règle `scope: allRows` exige que **tous** les lots soient conformes. Avec `scope: anyRow`, il suffirait qu'un seul le soit.

11.6.10 Bonnes pratiques des requêtes de supervision

Une requête de supervision s'exécute à intervalle régulier, en permanence. Elle doit être **légère** et **sans effet sur l'activité de production**.

Pratique	Raison
Requêtes agrégées renvoyant une seule ligne	Résultat toujours évaluable, transfert minimal
Filtrer sur une fenêtre de temps récente (`date > maintenant - 1 heure`) et sur des colonnes indexées	Éviter le parcours complet de grandes tables
Vérifier le plan d'exécution avec l'administrateur de la base avant la mise en production	Une requête anodine peut se révéler coûteuse sur une grande volumétrie
Interroger une réplique en lecture lorsqu'elle existe	Aucun impact sur la base principale
Fixer une limite de durée côté serveur (par exemple `statement_timeout` pour PostgreSQL), en plus du timeout du contrôle	Une requête interrompue par la sonde peut continuer à s'exécuter sur le serveur
Adapter la fréquence au rythme du phénomène observé	Un traitement quotidien n'a pas besoin d'être vérifié toutes les minutes
Ne jamais écrire : ni `INSERT`, `UPDATE`, `DELETE`, ni procédure modifiant des données	Sécurité et intégrité des données
Éviter `SELECT *` et ne lire que les colonnes utiles aux règles	Performance ; aucune donnée personnelle inutile dans les résultats

11.6.11 Correspondance avec l'ancien format des scénarios BDD

Les clients de la génération précédente de la solution décrivaient leurs contrôles de bases de données dans des fichiers texte structurés en sections. Ce format **n'est pas utilisé** par la sonde actuelle. Le tableau suivant permet de transposer manuellement un contrôle existant.

> → **Évolution prévue** — Un outil d'import des anciens fichiers de scénarios BDD est prévu dans une version ultérieure pour faciliter la migration. La transposition est manuelle.

Ancien format	Contrôle `database`
Section `[DATABASE]`, clé `HOST`	`database.host`
Clé `PORT`	`database.port`
Clé `BDD_TYPE` : `MYSQL`, `POSTGRESQL`, `SQLSERVER2008`, `SQLSERVER2016`, `ORACLE`	`database.engine` : `mysql`, `postgresql`, `sqlserver`, `sqlserver`, `oracle`
Clé `USERNAME`	`database.username`
Clé `PASSWORD` (en clair dans le fichier)	Secret créé sur la sonde (chapitre 9), référencé par `database.passwordSecret` et déclaré dans `secrets`
Section `[SQL]`	`database.query`
Section `[PROC]`	`database.procedure`
Section `[PARAM]`, lignes `nom;valeur`	`database.params` : `{ "nom": valeur }`
Section `[RULES]`, lignes `COLONNE;OPÉRATEUR;VALEUR`	`database.rules` : `{ "column": …, "operator": …, "value": … }`
Opérateurs `LT`, `LTE`, `GT`, `GTE`, `EQ`	`lt`, `lte`, `gt`, `gte`, `eq`

Ancien format	Contrôle `database`
Règle appliquée à chaque ligne du résultat	`scope: "allRows"` (valeur par défaut)

> **Information** — La version 2008 de SQL Server n'est plus prise en charge par son éditeur. La compatibilité des pilotes actuels avec cette version n'est pas garantie ; les versions prises en charge sont indiquées dans les notes de version.

> **Avertissement** — Les anciens fichiers contenaient les mots de passe en clair. Lors de la migration, supprimez ces fichiers des postes et des partages après transposition, et profitez-en pour renouveler les mots de passe concernés.

11.7 Indicateurs métier

11.7.1 Principe

Un **indicateur métier** est une valeur qui mesure l'activité ou la santé fonctionnelle d'une application : nombre de commandes de l'heure, montant encaissé, nombre d'inscriptions, messages en attente, délai depuis la dernière synchronisation. Contrairement à un état (OK ou en échec), un indicateur se **suit dans le temps** : sa courbe révèle les tendances, les ruptures et les cycles d'activité.

Le paramètre `exportValues` d'un contrôle `database` publie la valeur d'une ou plusieurs colonnes du résultat, à chaque exécution. Chaque entrée associe :

- `column` : la colonne du résultat ;
- `indicator` : le nom de l'indicateur, en minuscules, chiffres et tirets bas (`commandes_heure` , `messages_en_attente`).

La valeur est publiée sous la métrique `sva.indicator.value` , avec les attributs `sva.check.id` et `sva.indicator.name` (chapitre 18). Elle est également conservée dans l'historique de la sonde et affichée dans le détail des exécutions (chapitre 17). La valeur publiée est celle de la **première ligne** du résultat.

11.7.2 Règles de nommage

Règle	Exemple correct	À éviter
Nom fonctionnel, stable dans le temps	`commandes_heure`	`count_1`
Unité ou période explicite dans le nom	`montant_encaisse_jour_eur`	`montant`
Aucune valeur variable dans le nom	`commandes_heure`	`commandes_2025_11_19`
Dix indicateurs au plus par contrôle	—	Un indicateur par magasin, par produit… (cardinalité incontrôlée)

> **Conseil** — Pour suivre un indicateur par entité (par magasin, par canal de vente), créez un contrôle par entité **significative**, en nombre limité. La sonde n'est pas un entrepôt de données : les analyses fines par dimension relèvent des outils décisionnels de l'entreprise.

11.8 Détection d'anomalies

11.8.1 Seuils fixes

Dans la version actuelle, la détection d'anomalies repose sur des **seuils fixes**, exprimés par les règles des contrôles `database` et par les assertions des contrôles `http` et `api`. C'est une méthode simple, explicable et prévisible. Elle convient lorsque la valeur normale de l'indicateur est stable.

11.8.2 Tenir compte des cycles d'activité

L'activité d'une entreprise varie selon l'heure et le jour : un volume de commandes normal à 3 heures du matin serait alarmant à 11 heures. Trois techniques permettent d'adapter les seuils aux cycles, sans fonction d'apprentissage.

1. Plusieurs contrôles, plusieurs plages horaires. Créez un contrôle par période d'activité, chacun avec ses propres seuils et une planification `cron` limitée à sa période :

Contrôle	Planification (`cron`)	Seuil critique
Commandes, heures ouvrées	`*/15 9-18 * * 1-5`	Moins de 40 commandes par heure
Commandes, samedi en journée	`*/15 9-18 * * 6`	Moins de 25 commandes par heure
Commandes, soirée du lundi au samedi	`*/15 19-22 * * 1-6`	Moins de 15 commandes par heure

Les plans d'exécution offrent aussi des **fenêtres d'activité** (`activeWindows`), qui suspendent l'exécution en dehors des plages définies (chapitre 12).

2. Comparaison calculée dans la requête. La requête SQL compare elle-même la période en cours à une référence historique : même heure les semaines précédentes, moyenne glissante. La règle porte alors sur un **ratio**, indépendant du niveau absolu d'activité (§ 11.8.3).

3. Détection par la plateforme d'observabilité. Les indicateurs publiés (`sva.indicator.value`) peuvent alimenter les fonctions de détection d'anomalies de Datadog, de Dynatrace ou de Grafana (chapitre 18). La sonde fournit la mesure, la plateforme l'analyse.

> → **Évolution prévue** — Une détection d'anomalies intégrée à la sonde, fondée sur la comparaison automatique avec l'historique local (même créneau des semaines précédentes, écart en pourcentage), est prévue dans une version ultérieure. Elle n'est pas disponible actuellement.

11.8.3 Exemple : chute du nombre de commandes

Une boutique en ligne fictive constate qu'un incident chez un prestataire de paiement peut faire chuter les commandes sans qu'aucun serveur ne soit en panne. Le contrôle suivant compare, toutes les 15 minutes :

- le nombre de commandes validées au cours des **60 dernières minutes** ;
- la moyenne des commandes validées sur la **même fenêtre horaire** lors des **quatre semaines précédentes**.

JSON

```
{
  "id": "commandes-chute-activite",
  "name": "Boutique : chute anormale des commandes",
  "description": "Compare les commandes de la dernière heure à la moyenne des 4 semaines précédentes
    au même créneau (exemple fictif).",
  "type": "database",
  "tags": { "application": "boutique", "environment": "production" },
  "schedule": { "cron": "*/15 8-22 * * *", "timezone": "Europe/Paris" },
  "timeout": "30s",
```

```
  "secrets": ["DB_REPORTING_PASSWORD"],
  "database": {
    "engine": "postgresql",
    "host": "db-reporting.example.internal",
    "port": 5432,
    "database": "ventes",
    "username": "supervision_ro",
    "passwordSecret": "DB_REPORTING_PASSWORD",
    "query": "WITH courant AS (SELECT COUNT(*) AS nb FROM commandes WHERE statut = 'VALIDEE' AND
    est_test = false AND date_validation >= now() - interval '60 minutes'), reference AS (SELECT
    COUNT(*) / 4.0 AS nb FROM commandes c, generate_series(1, 4) AS s(semaine) WHERE c.statut =
    'VALIDEE' AND c.est_test = false AND c.date_validation >= now() - (s.semaine * interval '7 days')
    - interval '60 minutes' AND c.date_validation < now() - (s.semaine * interval '7 days')) SELECT
    courant.nb AS commandes_heure, reference.nb AS reference_heure, CASE WHEN reference.nb < 10 THEN
    1.0 ELSE round(courant.nb / reference.nb, 2) END AS ratio FROM courant, reference",
    "rules": [
      { "column": "ratio", "operator": "gte", "value": 0.7, "severity": "warning", "scope":
    "firstRow" },
      { "column": "ratio", "operator": "gte", "value": 0.4, "severity": "critical", "scope":
    "firstRow" }
    ],
    "exportValues": [
      { "column": "commandes_heure", "indicator": "commandes_heure" },
      { "column": "reference_heure", "indicator": "commandes_heure_reference" },
      { "column": "ratio", "indicator": "commandes_ratio_reference" }
    ]
  }
}
```

Fonctionnement :

Élément	Rôle
`courant`	Commandes validées des 60 dernières minutes, commandes de test exclues (`est_test = false`)
`reference`	Moyenne, sur les 4 semaines précédentes, des commandes validées dans la même fenêtre de 60 minutes
`ratio`	Activité actuelle rapportée à la référence : 1,0 = activité habituelle ; 0,5 = activité réduite de moitié
`CASE WHEN reference.nb < 10 THEN 1.0`	Neutralise le contrôle lorsque l'activité de référence est trop faible pour être significative (nuit, jour férié) : un ratio calculé sur quelques commandes produirait des fausses alertes
Règle `warning`	Activité inférieure à 70 % de la normale
Règle `critical`	Activité inférieure à 40 % de la normale
`exportValues`	Publie les trois valeurs : les courbes `commandes_heure` et `commandes_heure_reference` se superposent dans un tableau de bord et rendent l'écart immédiatement lisible

Résultat d'une exécution lors d'un incident :

```json
{
  "checkId": "commandes-chute-activite",
  "status": "CRITICAL",
  "durationMs": 412,
  "error": {
    "type": "assertion_failed",
    "message": "Règle ratio gte 0.4 (critical) non respectée : valeur 0.31"
  },
  "measurements": {
    "indicator.commandes_heure": 22,
    "indicator.commandes_heure_reference": 71.5,
    "indicator.commandes_ratio_reference": 0.31
  }
}
```

Attention — Les jours atypiques (soldes, jours fériés, campagnes promotionnelles, panne lors d'une des semaines de référence) faussent la référence. Prévoyez de suspendre le contrôle ou d'ajuster ses seuils lors de ces périodes, et informez les équipes métier de son mode de calcul. Faites valider la requête par l'administrateur de la base : son coût dépend de l'indexation de la colonne de date.

11.9 Précautions et sécurité

Les contrôles applicatifs interagissent avec des applications de production et des données de l'entreprise. Ils exigent davantage de précautions que les contrôles techniques.

11.9.1 Ne jamais modifier de données de production

Règle	Application
Lecture seule par défaut	Requêtes `GET`, recherches, consultations ; comptes en lecture seule
Aucune transaction réelle	Pas de commande, de paiement, de réservation, de virement, d'envoi de message à un client réel
Opérations d'écriture uniquement en environnement de test	Création-lecture-suppression d'objets de test sur une plateforme de recette, avec des données fictives
Données témoins identifiées	Marquage des objets de test pour les exclure des statistiques, de la facturation et des traitements métier

Avertissement — Un contrôle qui écrit des données s'exécute des centaines de fois par jour. Une erreur de conception peut créer des milliers d'objets, déclencher des envois de courriers électroniques ou de SMS à de vrais clients, ou fausser la comptabilité. Toute exception à la lecture seule doit être validée par le responsable de l'application et limitée à un environnement de test.

11.9.2 Protéger les identifiants

- Stockez tous les mots de passe, clés d'API et secrets clients comme **secrets** de la sonde (chapitre 9). Ne les écrivez jamais en clair dans une URL, un en-tête ou une requête.
- Déclarez dans chaque contrôle **uniquement** les secrets dont il a besoin.

- Limitez la liste `allowedHosts` de chaque secret aux seuls hôtes qui doivent le recevoir (chapitre 9, § 9.8.1) : serveur d'autorisation, API, serveur de base de données. Un secret n'est jamais injecté vers un hôte absent de cette liste, ni retransmis après une redirection vers un autre hôte.
- Utilisez des comptes de supervision **dédiés**, aux droits minimaux **vérifiés côté serveur** (§ 11.6.4), et renouvelez leurs secrets selon la politique de l'entreprise.

11.9.3 Maîtriser l'impact sur les applications

- Adaptez la fréquence au besoin. Un contrôle toutes les minutes sur une API coûteuse peut représenter une charge significative.
- Prévenez les équipes applicatives et sécurité de la mise en place des contrôles. Les requêtes de la sonde ne doivent pas être prises pour une attaque par les pare-feu applicatifs ou les outils de détection d'intrusion.
- Identifiez le trafic de supervision (en-tête `User-Agent` explicite, compte dédié) pour qu'il puisse être exclu des statistiques d'audience et d'activité.
- Respectez les quotas des API de partenaires : certaines facturent chaque appel.

11.9.4 Limiter les données collectées

Les résultats des contrôles sont stockés sur la sonde et peuvent être exportés vers des plateformes tierces. N'interrogez et ne vérifiez que les données strictement nécessaires. Évitez toute donnée personnelle réelle dans les requêtes, les données témoins et les assertions.

11.10 Synthèse

Besoin	Contrôle	Édition	Section
Réponse correcte d'une API (une requête)	`http` + `jsonPath`	Essential	§ 11.2
Authentification puis appel d'une API	`api`	Professional	§ 11.3
Web service SOAP	`http` + `xpath` (ou `api`)	Essential (Professional pour `api`)	§ 11.4
Traitements planifiés, files, cohérence de données	`database`	Professional	§ 11.6
Suivi d'un indicateur d'activité	`database` + `exportValues`	Professional	§ 11.7
Chute ou pic d'activité	`database` avec ratio calculé	Professional	§ 11.8
Parcours utilisateur dans un navigateur	`browser`	Enterprise	Partie IV

Chapitre 12 — Planification et plans d'exécution

Ce chapitre explique comment organiser dans le temps l'exécution des contrôles, par deux mécanismes :

- la **planification individuelle** de chaque contrôle ;
- les **plans d'exécution** (*playlists*), qui enchaînent plusieurs contrôles dans un ordre déterminé.

Il montre aussi comment évaluer la charge d'une sonde, pour garantir que les contrôles s'exécutent à l'heure prévue.

Les notions de fréquence, de gigue, de timeout et de concurrence sont introduites au chapitre 8. Ce chapitre en détaille la mise en œuvre. Les exemples utilisent des applications et des noms de domaine fictifs.

12.1 Deux façons d'organiser les exécutions

	Planification individuelle	Plan d'exécution
Porte sur	Un contrôle	Une liste ordonnée de contrôles
Définie dans	La propriété `schedule` du contrôle	Un objet `plan` distinct
Ordre d'exécution	Indépendant des autres contrôles	Séquentiel, dans l'ordre de la liste
Dépendances	Aucune	Arrêt possible au premier échec (`stopOnFailure`)
Plages horaires	Par expression cron	Par expression cron et plages actives (`activeWindows`)
Usage typique	Disponibilité d'un service, contrôles réseau, indicateurs métier	Parcours enchaînés, scénarios dépendants, maîtrise de la charge du moteur de scénarios

Les deux mécanismes se combinent : un contrôle peut avoir sa propre planification **et** figurer dans un ou plusieurs plans. Il peut aussi n'être exécuté que par un plan.

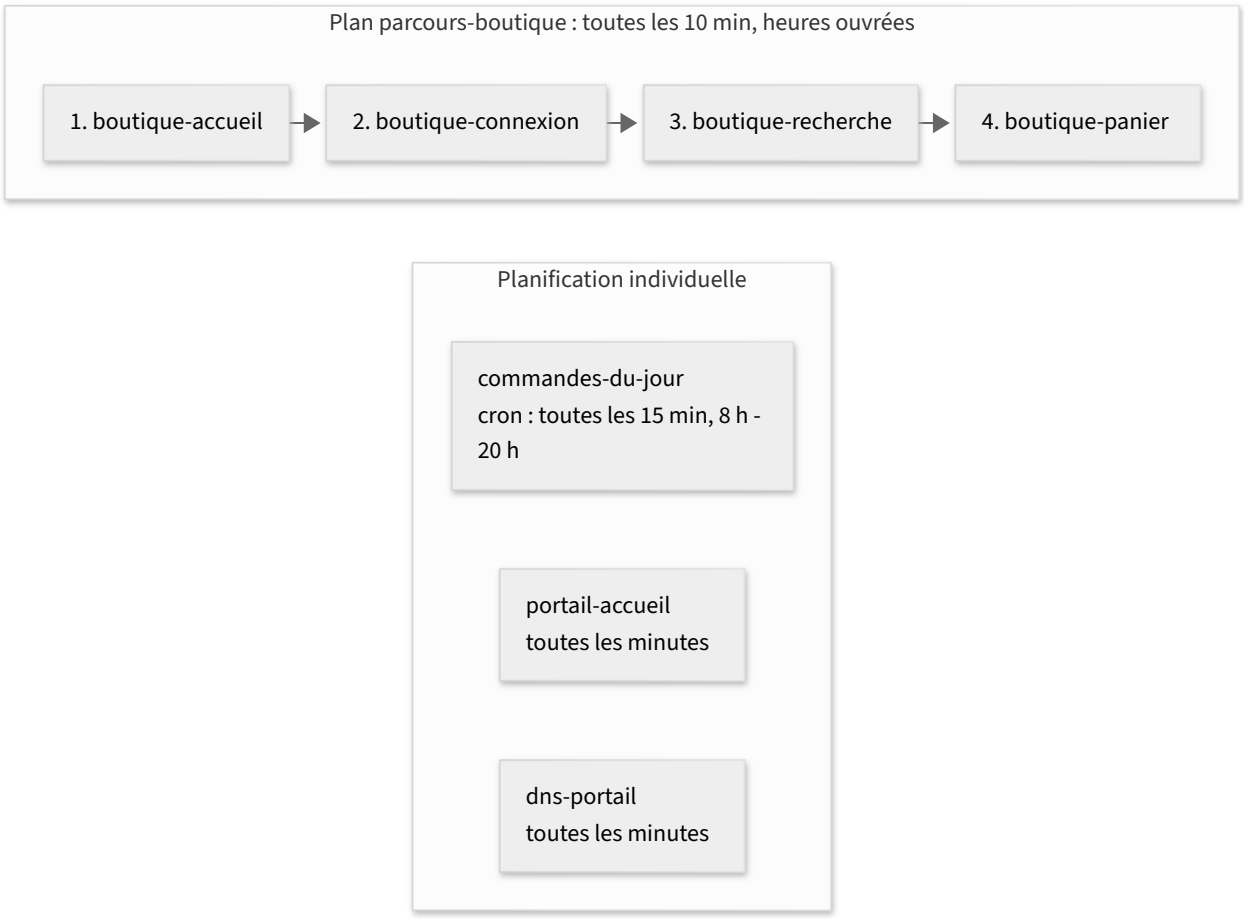


Schéma 12-1 — Planification individuelle et plan d'exécution

12.2 Planifier un contrôle

12.2.1 Par intervalle

L'intervalle est le mode le plus simple et le plus courant : le contrôle s'exécute à intervalles réguliers, en continu.

```json
"schedule": { "interval": "5m", "jitter": "20s" }
```

Propriété	Rôle	Format
`interval`	Écart entre deux exécutions successives	Durée : `30s`, `1m`, `5m`, `1h`
`jitter`	Décalage aléatoire maximal ajouté à chaque échéance (chapitre 8, § 8.3.2)	Durée

Les intervalles sont comptés d'un **début d'exécution** au suivant. Si une exécution dure encore au moment de l'échéance suivante, cette échéance est **sautée** : un même contrôle ne s'exécute jamais deux fois en parallèle. Une échéance sautée ne produit aucune exécution : elle se repère dans l'historique du contrôle, et une saturation durable se mesure par le retard de planification (métrique `sva.agent.scheduler.lag`, chapitre 18).

12.2.2 Par expression cron

Une **expression cron** décrit des instants d'exécution selon le calendrier. C'est le mode adapté aux contrôles qui n'ont de sens qu'à certains moments : heures ouvrées, fin d'un traitement de nuit, ouverture d'un service.

```json
"schedule": { "cron": "*/15 8-20 * * 1-5", "timezone": "Europe/Paris" }
```

La sonde utilise le format cron standard à **cinq champs**, sans champ des secondes :

Position	Champ	Valeurs	Exemple
1	Minute	0-59	`*/15` : toutes les 15 minutes
2	Heure	0-23	`8-20` : de 8 h à 20 h inclus
3	Jour du mois	1-31	`*` : tous les jours
4	Mois	1-12	`*` : tous les mois
5	Jour de la semaine	0-7 (0 et 7 = dimanche, 1 = lundi)	`1-5` : du lundi au vendredi

Opérateurs : `*` (toutes les valeurs), `,` (liste), `-` (plage), `/` (pas).

Expression	Signification
`*/5 * * * *`	Toutes les 5 minutes
`0 * * * *`	Toutes les heures, à l'heure pile
`*/15 8-20 * * 1-5`	Toutes les 15 minutes de 8 h 00 à 20 h 45, du lundi au vendredi
`30 6 * * *`	Chaque jour à 6 h 30, par exemple après un traitement de nuit
`0 9,14 * * 1-5`	À 9 h et 14 h, les jours ouvrés
`0 7 1 * *`	Le premier jour de chaque mois à 7 h

> **Information** — Dans la plage `8-20` du champ des heures, la dernière exécution d'une expression `*/15 8-20 * * *` a lieu à **20 h 45**, et non à 20 h 00 : le champ des heures inclut toute l'heure 20. Pour arrêter à 20 h 00 exactement, combinez deux planifications, ou utilisez les plages actives d'un plan (§ 12.4.4).

Une gigue (`jitter`) peut aussi s'appliquer à une planification cron. Elle évite que tous les contrôles planifiés « à l'heure pile » démarrent simultanément.

12.2.3 Fuseau horaire et changement d'heure

La propriété `timezone` indique le fuseau horaire dans lequel l'expression cron est interprétée, sous la forme d'un identifiant de la base IANA (`Europe/Paris`, `America/Montreal`, `UTC`). Sans indication, l'expression est interprétée en **UTC**.

> **Attention** — Une expression cron interprétée en UTC décale vos plages horaires d'une ou deux heures par rapport à l'heure de Paris, selon la saison. Indiquez toujours le fuseau horaire des utilisateurs de l'application supervisée.

Lors des changements d'heure, certaines heures locales n'existent pas (au printemps) ou existent deux fois (à l'automne). Une heure planifiée qui n'existe pas déclenche l'exécution à la première minute valide suivante ; une heure qui existe deux fois ne produit qu'une seule exécution. Évitez de planifier des contrôles critiques entre 2 h et 3 h

du matin dans les fuseaux soumis à l'heure d'été.

Les règles détaillées appliquées lors des deux bascules sont les suivantes, dans cet ordre :

1. **Calcul des échéances** dans le fuseau `timezone` de la planification (contrôle ou plan).
2. **Heure inexistante** (bascule de printemps) : l'échéance est reportée à la première minute valide suivante.
3. **Heure répétée** (bascule d'automne) : l'échéance ne donne lieu qu'à une exécution, lors de la **première occurrence** de l'heure répétée (heure d'été).
4. **Déduplication** : plusieurs échéances ramenées à la même minute ne donnent qu'**une seule exécution**.
5. **Gigue** : le décalage aléatoire (`jitter`) s'applique ensuite, à cette exécution unique.

Exemple, dans le fuseau `Europe/Paris`, la nuit de la bascule de printemps (de 2 h 00 à 3 h 00) : avec l'expression `*/30 2-3 * * *`, les échéances de 2 h 00 et de 2 h 30 n'existent pas et sont reportées à 3 h 00, où se trouve déjà une échéance. Le contrôle s'exécute **une seule fois** à 3 h 00 (plus la gigue), puis à 3 h 30.

Plages actives et fenêtres lors des bascules. Les bornes des plages actives d'un plan (§ 12.4.4) et l'heure de début des fenêtres de maintenance des mises à jour (chapitre 20) sont des heures **locales**, exprimées dans leur fuseau :

Situation	Effet
Fenêtre de maintenance qui commence dans l'heure inexistante	Elle commence à la première minute valide suivante
Fenêtre de maintenance qui commence dans l'heure répétée	Elle n'a lieu qu'une fois, à la **première occurrence** de l'heure répétée (heure d'été)
Plage active qui franchit minuit et contient la bascule (par exemple de 22:00 à 04:00, la nuit du samedi au dimanche du changement d'heure)	Ses bornes restent fixes en heure locale : elle dure une heure de moins en temps réel au printemps, une heure de plus à l'automne
Fenêtre de maintenance qui contient la bascule	Une fenêtre est définie par un début local et une durée (`windows[].start` , `windows[].duration`), et non par deux bornes. La durée est un temps écoulé depuis le début effectif, et la fin locale se décale d'une heure cette nuit-là (chapitre 20, § 20.4.8)
Plage active dont une borne tombe dans l'heure inexistante ou répétée	Même principe que pour les échéances : au printemps, la borne est reportée à la première minute valide suivante ; à l'automne, elle est interprétée à sa **première occurrence** (heure d'été). Une plage qui se termine dans l'heure répétée prend donc fin lors de son premier passage

Correspondance avec les outils de mise en sourdine. Lorsque vous reproduisez une plage ou une fenêtre de la sonde dans un outil externe (intervalles de temps d'Alertmanager, temps d'arrêt programmés, périodes de maintenance), configurez-le dans **le même fuseau horaire IANA** que la sonde, et non en UTC ou dans un décalage fixe comme `+01:00` : sinon, les deux périodes se décalent d'une heure à chaque bascule. Le même fuseau est nécessaire mais ne suffit pas à garantir la même résolution des heures ambiguës : chaque outil applique ses propres règles à une heure inexistante ou répétée, et un outil défini par deux bornes civiles ne calcule pas nécessairement la fin d'une fenêtre comme la sonde. Vérifiez donc, sur la première bascule suivant la mise en service, que la sourdine couvre effectivement la période de la sonde, ou évitez de placer les bornes entre 2 h et 3 h du matin.

Les horodatages des résultats restent enregistrés en UTC, quel que soit le fuseau de planification. L'interface d'administration les affiche dans le fuseau configuré dans les paramètres `time` (chapitre 9, § 9.9).

12.2.4 Choisir entre intervalle et cron

Besoin	Mode recommandé
Surveiller en continu la disponibilité d'un service	Intervalle

Besoin	Mode recommandé
Contrôler uniquement pendant les heures d'ouverture d'un service	Cron (ou plan avec plages actives)
Vérifier le résultat d'un traitement planifié (import nocturne, clôture)	Cron, après l'heure de fin prévue du traitement
Suivre un indicateur métier à rythme régulier	Cron, aligné sur l'heure pile ou le quart d'heure pour faciliter la lecture des courbes
Contrôler un environnement de recette disponible en journée	Cron

12.2.5 Valeurs recommandées

Type	Intervalle conseillé	Gigue conseillée	Timeout conseillé
`icmp`, `tcp`, `dns`	1 min	10 s	5 à 10 s
`http`	1 à 5 min	10 s pour 1 min, 30 s pour 5 min	10 à 30 s
`api`	2 à 10 min	30 s	30 à 60 s
`nagios`	1 à 5 min	10 s pour 1 min, 30 s pour 5 min	Selon le plugin
`database`	5 à 15 min ou cron	30 s	30 à 60 s
`browser`	5 à 15 min, de préférence au sein d'un plan	30 s	60 à 180 s

Une règle simple s'applique à tous les types : la gigue doit rester nettement inférieure à l'intervalle (de l'ordre de 10 à 25 % au plus), et le timeout inférieur à l'intervalle.

12.2.6 Contrôle sans planification propre

Un contrôle dont la planification vaut `null` n'est jamais lancé de lui-même. Il n'est exécuté que :

- au sein d'un plan d'exécution ;
- manuellement, depuis l'interface ou par `POST /api/v1/checks/{id}/runs`.

C'est la configuration habituelle des contrôles qui n'ont de sens qu'à l'intérieur d'un parcours enchaîné, comme l'ajout au panier, qui suppose une connexion préalable.

```json
{
  "id": "boutique-panier",
  "name": "Boutique : ajout d'un article témoin au panier (recette)",
  "type": "browser",
  "schedule": null,
  "timeout": "90s",
  "secrets": ["BOUTIQUE_USER", "BOUTIQUE_PASSWORD"],
  "browser": {
    "mode": "script",
    "scriptRef": "boutique-panier@sha256:9a7e31c4d2b8",
    "params": { "baseUrl": "https://recette.boutique.example.com" }
  }
}
```

12.3 Les plans d'exécution

12.3.1 Intérêt

Un **plan d'exécution**, ou *playlist*, est une liste ordonnée de contrôles exécutés **l'un après l'autre**, selon une planification propre au plan. Il répond à trois besoins.

Besoin	Apport du plan
Enchaîner des contrôles dépendants	Inutile de vérifier le panier si la connexion a échoué : avec `stopOnFailure`, le plan s'arrête au premier échec, et l'alerte porte sur la vraie cause
Maîtriser la charge	Les scénarios navigateur sont coûteux et la sonde n'en exécute qu'un à la fois par défaut. Les regrouper dans un plan garantit une exécution ordonnée, sans accumulation dans la file d'attente
Reproduire un parcours de bout en bout	La séquence des contrôles reflète l'ordre réel des actions des utilisateurs : accueil, connexion, recherche, panier

12.3.2 Structure d'un plan

Un plan est décrit par un objet JSON dont la structure est validée par le schéma `plan.schema.json` de la spécification.

Propriété	Obligatoire	Rôle	Défaut
`id`	Oui	Identifiant stable (mêmes règles que pour un contrôle)	—
`name`	Oui	Libellé lisible	—
`enabled`	Non	Plan actif ou suspendu	`true`
`schedule`	Oui	Planification du plan : `interval` ou `cron`, `timezone`, `jitter` (§ 12.2)	—
`activeWindows`	Non	Plages horaires pendant lesquelles le plan s'exécute	Toujours
`stopOnFailure`	Non	Interrompre le plan au premier contrôle en échec	`false`
`maxDuration`	Non	Durée maximale de l'ensemble du plan	`30m`
`items`	Oui	Liste ordonnée des contrôles (1 à 50)	—

Chaque élément de `items` désigne un contrôle par son identifiant (`checkId`). Il peut aussi préciser `continueOnFailure`, qui déroge localement à `stopOnFailure` (§ 12.4.2).

Exemple :

```json
{
  "id": "parcours-boutique",
  "name": "Parcours complet boutique (recette)",
  "enabled": true,
  "schedule": { "cron": "*/10 * * * *", "timezone": "Europe/Paris", "jitter": "30s" },
  "activeWindows": [
    { "days": ["mon", "tue", "wed", "thu", "fri"], "from": "07:00", "to": "21:00" }
  ],
  "stopOnFailure": true,
  "maxDuration": "8m",
  "items": [
    { "checkId": "boutique-accueil" },
    { "checkId": "boutique-connexion" },
    { "checkId": "boutique-recherche", "continueOnFailure": true },
    { "checkId": "boutique-panier" }
  ]
}
```

Ce plan s'exécute toutes les 10 minutes, du lundi au vendredi de 7 h à 21 h, heure de Paris. Il enchaîne quatre scénarios et s'arrête au premier échec, sauf pour la recherche : un échec de la recherche n'empêche pas de vérifier le panier.

12.4 Comportement d'un plan

12.4.1 Déroulement

À chaque échéance de sa planification, et si l'on se trouve dans une plage active, le plan :

1. vérifie qu'aucune exécution précédente du même plan n'est encore en cours. Si c'est le cas, l'échéance est **sautée** ;
2. exécute le premier contrôle de la liste et attend sa fin, **nouvelles tentatives comprises** ;
3. passe au contrôle suivant, ou s'arrête selon les règles de `stopOnFailure` (§ 12.4.2) ;
4. se termine lorsque tous les contrôles ont été traités, que le plan a été interrompu par un échec, ou que la durée maximale `maxDuration` est atteinte (§ 12.4.3).

Chaque contrôle exécuté par un plan produit un résultat ordinaire, avec ses propres état, durée, étapes et erreur. Ce résultat porte le type de déclenchement `plan` et un identifiant d'exécution de plan (`planRunId`). Celui-ci permet de retrouver tous les résultats d'un même passage. Chaque contrôle met à jour son propre état courant, exactement comme s'il avait été lancé seul.

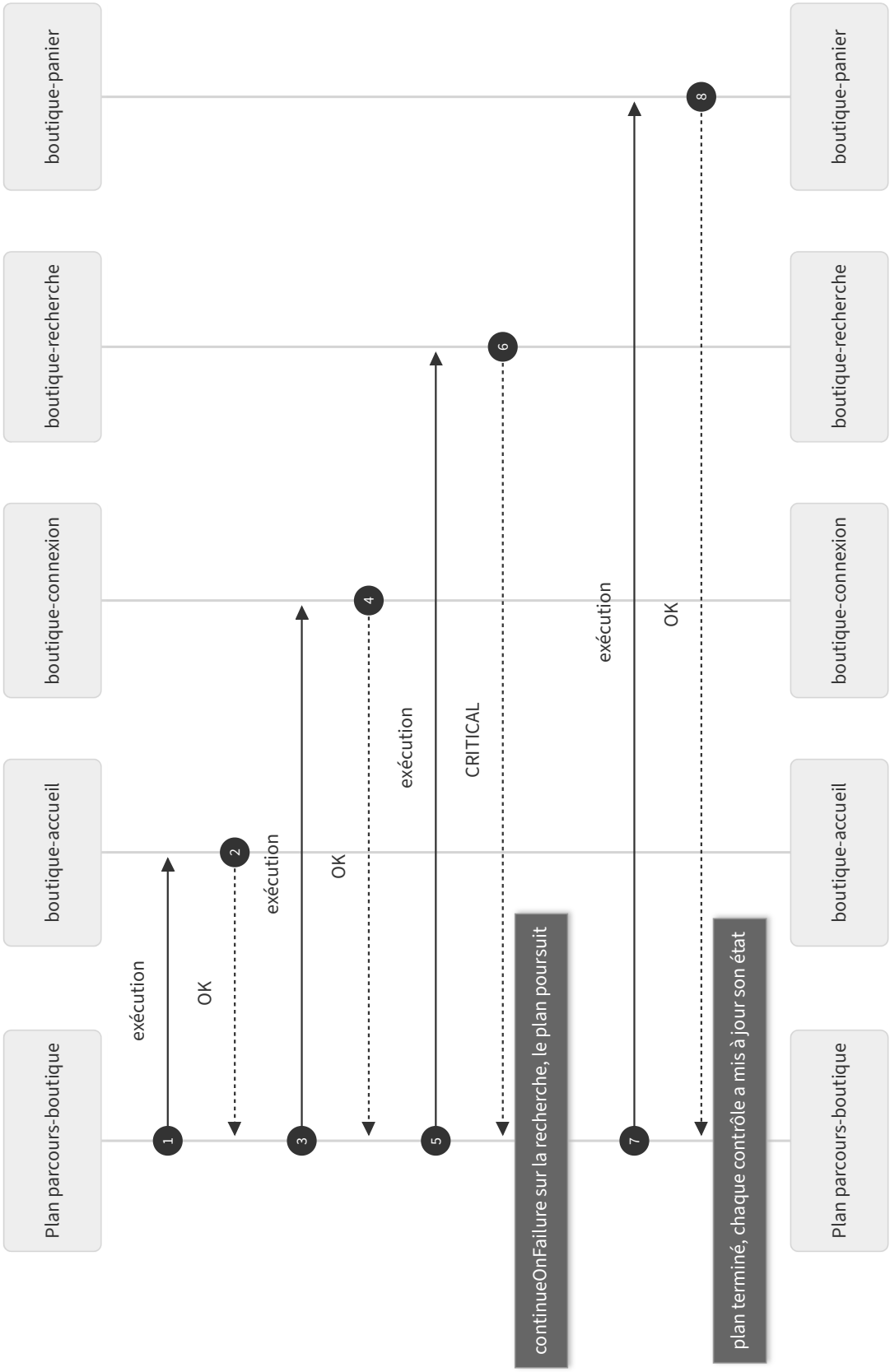


Schéma 12-2 — Déroulement d'un plan d'exécution

Schéma présenté en orientation paysage : tournez le document d'un quart de tour pour le lire.

12.4.2 Arrêt sur échec : stopOnFailure et continueOnFailure

`stopOnFailure` du plan	`continueOnFailure` de l'élément	Comportement si l'élément est en échec
`false` (défaut)	absent	Le plan continue
`false`	`false`	Le plan s'arrête
`true`	absent	Le plan s'arrête
`true`	`true`	Le plan continue

Est considéré comme un **échec** un état final `CRITICAL` ou `UNKNOWN`, après nouvelles tentatives. Un état `WARNING` (réussite dégradée) n'interrompt pas le plan.

Les contrôles non exécutés à cause d'un arrêt ne produisent aucun résultat : leur état courant reste celui de leur dernière exécution.

> ✓ **Conseil** — Activez `stopOnFailure` lorsque les contrôles suivants dépendent réellement du précédent (connexion, puis actions authentifiées). Utilisez `continueOnFailure` pour les vérifications indépendantes insérées dans le parcours, comme la recherche ou une page d'information.

> ! **Attention** — Lorsqu'un plan s'arrête sur échec, les contrôles situés après le point d'arrêt ne sont plus vérifiés tant que l'échec persiste. Si l'un d'eux n'est exécuté que par le plan (planification `null`), son état peut devenir `STALE` (§ 12.4.5). C'est le comportement attendu : il indique que cette partie du parcours n'a pas pu être évaluée.

12.4.3 Durée maximale du plan

`maxDuration` borne la durée totale d'un passage. Lorsqu'elle est atteinte, le contrôle en cours est mené à son terme dans la limite de son propre timeout, puis le plan s'arrête. Les contrôles restants ne sont pas exécutés.

Fixez `maxDuration` à une valeur comprise entre :

- la somme des timeouts des contrôles du plan, si vous voulez qu'aucun contrôle ne soit privé d'exécution ;
- l'intervalle du plan, pour qu'un passage se termine avant l'échéance suivante.

Si la première valeur dépasse la seconde, le plan est trop chargé pour sa fréquence (§ 12.6).

12.4.4 Plages actives

Les **plages actives** (`activeWindows`) limitent l'exécution d'un plan à certains jours et horaires, exprimés dans le fuseau horaire de sa planification.

```json
"activeWindows": [
  { "days": ["mon", "tue", "wed", "thu", "fri"], "from": "07:00", "to": "21:00" },
  { "days": ["sat"], "from": "09:00", "to": "13:00" }
]
```

Propriété	Valeurs
`days`	`mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun`
`from`, `to`	Heure au format `HH:MM` (00:00 à 23:59)

Un plan peut compter jusqu'à 14 plages. En dehors de toute plage, les échéances du plan sont ignorées. Une plage peut franchir minuit : une heure de début postérieure à l'heure de fin, par exemple de 22:00 à 02:00, indique un franchissement de minuit. Le comportement des plages lors des changements d'heure est décrit au § 12.2.3.

Les plages actives servent par exemple à :

- superviser un environnement de recette uniquement pendant les heures où il est disponible ;
- ne pas générer de trafic synthétique pendant les traitements de nuit d'une application ;
- adapter la supervision aux horaires d'ouverture d'un service.

12.4.5 Plans, états et STALE

Un contrôle est déclaré `STALE` lorsqu'il ne produit aucun résultat pendant plus de deux intervalles de sa planification (chapitre 8, § 8.8.4). Pour un contrôle exécuté uniquement par un plan, la référence est la planification **du plan**. En dehors des plages actives, aucun contrôle n'est déclaré `STALE` pour cause d'absence d'exécution.

12.4.6 Un contrôle dans plusieurs plans

Un même contrôle peut figurer dans plusieurs plans, par exemple un scénario de connexion partagé entre deux parcours. Chaque exécution met à jour l'état courant du contrôle, quel que soit le plan à l'origine. Si deux plans lancent le même contrôle simultanément, la seconde demande est ignorée tant que la première n'est pas terminée.

> ✓ **Conseil** — Évitez de multiplier les plans contenant les mêmes scénarios navigateur : chaque passage consomme du temps de moteur de scénarios. Un parcours unique, bien construit, est généralement préférable à plusieurs plans redondants.

12.5 Créer et piloter un plan

12.5.1 Opérations disponibles

Opération	Méthode et chemin	Rôle minimal
Lister les plans	`GET /api/v1/plans`	`viewer`
Lire un plan	`GET /api/v1/plans/{id}`	`viewer`
Créer	`POST /api/v1/plans`	`operator`
Remplacer	`PUT /api/v1/plans/{id}`	`operator`
Modifier partiellement	`PATCH /api/v1/plans/{id}`	`operator`
Activer, désactiver	`POST /api/v1/plans/{id}/enable`, `POST /api/v1/plans/{id}/disable`	`operator`
Lister les résultats d'un passage de plan	`GET /api/v1/runs?planRunId=…`	`viewer`
Supprimer	`DELETE /api/v1/plans/{id}`	`operator`
Exécuter immédiatement	`POST /api/v1/plans/{id}/runs`	`operator`

Les principes généraux de l'API (authentification, ETag et `If-Match`, format des erreurs) sont décrits au chapitre 9. Les exemples réutilisent la fonction `sapi` définie au chapitre 9, § 9.3.3.

12.5.2 Procédure de création

Objectif : mettre en place un plan qui enchaîne quatre scénarios de la boutique de recette toutes les 10 minutes, en heures ouvrées.

Prérequis :

- rôle `operator` ;
- édition Enterprise, puisque le plan contient des scénarios navigateur ;
- les quatre contrôles existent et ont été validés individuellement par une exécution d'essai (chapitre 9, § 9.6.5) ;
- les contrôles destinés à n'être exécutés que par le plan ont une planification `null` (§ 12.2.6).

Étapes :

1. Rédigez la définition du plan dans un fichier `parcours-boutique.json` (exemple du § 12.3.2). Pour la première mise en service, indiquez `"enabled": false`.
2. Créez le plan :

```bash
sapi -X POST -H 'Content-Type: application/json' \
  --data @parcours-boutique.json "$SONDE/api/v1/plans"
```

3. Lancez une exécution immédiate et observez le déroulement (§ 12.5.3).
4. Si le résultat est conforme, activez le plan : `POST /api/v1/plans/{id}/enable`.

Résultat attendu : le plan apparaît dans la liste ; ses exécutions sont visibles dans l'historique de chacun des contrôles, avec le déclenchement `plan`.

Vérification : après au moins deux échéances, vérifiez dans l'historique des exécutions (chapitre 17) que les quatre contrôles sont exécutés dans l'ordre, et que la durée totale d'un passage reste nettement inférieure à 10 minutes.

En cas de problème :

- réponse `422` : un identifiant de `items` ne désigne aucun contrôle existant, ou la planification est invalide ;
- plan qui ne s'exécute jamais : vérifiez les plages actives et le fuseau horaire.

12.5.3 Exécution immédiate d'un plan

L'exécution immédiate lance un passage complet du plan, sans attendre son échéance. Elle sert à valider un plan après sa création ou sa modification, et à confirmer qu'un parcours est rétabli après un incident.

```bash
sapi -X POST -D - "$SONDE/api/v1/plans/parcours-boutique/runs"
```

La sonde répond `202 Accepted`. Le déroulement se suit dans l'interface d'administration, par le flux d'événements en temps réel (chapitre 9, § 9.11) ou en consultant l'historique des contrôles concernés. L'exécution immédiate d'un plan désactivé, ou en dehors de ses plages actives, est autorisée.

12.5.4 Modifier l'ordre ou le contenu d'un plan

La modification d'un plan (`PUT` ou `PATCH`) prend effet au **prochain passage**. Un passage en cours se poursuit avec l'ancienne définition.

Retirer un contrôle d'un plan ne supprime pas le contrôle. S'il n'a pas de planification propre, il n'est plus exécuté du tout : pensez à lui en attribuer une, ou à le désactiver, pour que la situation reste explicite.

12.6 Charge et dimensionnement

12.6.1 Pourquoi calculer la charge

Chaque classe d'exécution dispose d'un nombre limité d'exécutions simultanées (chapitre 8, § 8.11) : 16 pour les contrôles légers, 4 pour les plugins et les bases de données, **une seule** par défaut pour les scénarios navigateur. Si la charge demandée dépasse la capacité d'une classe, les exécutions s'accumulent dans la file d'attente. Leur retard grandit, et les contrôles finissent par devenir `STALE`.

La classe navigateur est de loin la plus exposée. Avant de multiplier les scénarios, il convient donc de calculer la charge.

12.6.2 Méthode de calcul

Pour une classe d'exécution, le **taux d'occupation** se calcule ainsi :

```
TEXTE
Taux d'occupation = Somme, pour chaque contrôle, de (durée moyenne × exécutions par heure)
                    ──────────────────────────────────────────────────────────────────────
                    3 600 secondes × concurrence de la classe
```

Le nombre d'exécutions par heure vaut 3 600 divisé par l'intervalle en secondes. Pour un contrôle cron, comptez les exécutions de l'heure la plus chargée.

Taux d'occupation	Appréciation
Moins de 50 %	Confortable : marge suffisante pour les nouvelles tentatives, les exécutions manuelles et les ralentissements
50 à 70 %	Acceptable ; à surveiller
Plus de 70 %	Risqué : la moindre série de nouvelles tentatives ou de timeouts crée des retards
Plus de 100 %	Saturé : des exécutions sont retardées en permanence, puis des contrôles deviennent `STALE`

> **Information** — Utilisez les durées réellement observées sur la sonde (chapitre 17), et non celles mesurées sur un poste de travail. Comptez aussi les **nouvelles tentatives** : un scénario en échec avec une nouvelle tentative consomme deux fois son temps d'exécution, et un timeout consomme la totalité de la durée maximale.

12.6.3 Exemple chiffré

Une sonde Enterprise (concurrence navigateur : 1) doit exécuter six scénarios navigateur, chacun d'une durée moyenne de 45 secondes.

Configuration initiale : chaque scénario est planifié individuellement toutes les 5 minutes, soit 12 exécutions par heure.

```
TEXTE
Charge = 6 scénarios × 45 s × 12 exécutions/h = 3 240 s par heure
Taux   = 3 240 / (3 600 × 1) = 90 %
```

Le taux de 90 % est trop élevé. Il suffit que deux scénarios échouent par timeout (par exemple 120 secondes chacun, nouvelle tentative comprise) pour saturer la file pendant plusieurs minutes. Les exécutions se décalent, et l'état de certains scénarios devient `STALE`.

Configuration corrigée : les six scénarios sont regroupés dans un plan exécuté toutes les 10 minutes.

```
TEXTE
Durée d'un passage = 6 × 45 s = 270 s (4 min 30 s)
Charge             = 270 s × 6 passages/h = 1 620 s par heure
Taux               = 1 620 / 3 600 = 45 %
```

Le taux de 45 % laisse une marge confortable. La durée d'un passage, 4 min 30 s, reste inférieure à l'intervalle de 10 minutes : une `maxDuration` de 8 minutes est cohérente.

Alternatives si une fréquence de 5 minutes est indispensable pour certains parcours :

Option	Effet
Ne conserver à 5 minutes que les deux parcours les plus critiques ; passer les autres à 15 minutes	Taux de 50 % : 2 × 45 × 12 + 4 × 45 × 4 = 1 800 s par heure
Optimiser les scénarios (chapitre 16) : supprimer les attentes inutiles, bloquer les ressources tierces	Durée moyenne réduite, donc charge réduite
Porter la concurrence navigateur à 2 sur une sonde de 8 Go	Capacité doublée, au prix d'une consommation mémoire accrue (chapitre 8, § 8.11.2). Valeur indicative
Répartir les scénarios sur une seconde sonde	Capacité doublée et vision depuis un second site

12.6.4 Contrôles légers

Pour les contrôles légers, la charge est rarement un problème. Exemple : 60 contrôles `http` toutes les minutes, d'une durée moyenne de 0,8 seconde.

```
TEXTE
Charge = 60 × 0,8 s × 60 exécutions/h = 2 880 s par heure
Taux   = 2 880 / (3 600 × 16) = 5 %
```

La vigilance porte alors davantage sur le **volume de données** produit (chapitre 8, § 8.12.2) et sur la charge imposée aux cibles que sur la capacité de la sonde.

12.6.5 Surveiller la charge dans la durée

Le calcul initial doit être confirmé par l'observation. La sonde publie des métriques de santé (chapitre 18) qui permettent de suivre :

- le **retard de planification** par classe d'exécution : l'écart entre l'heure prévue et l'heure effective de démarrage ;
- la **profondeur des files d'attente** par classe ;
- les **échéances sautées**, visibles dans l'historique des contrôles (intervalles sans exécution).

Un retard de planification qui augmente régulièrement, ou des échéances sautées fréquentes, imposent de revoir la planification selon les options du § 12.6.3.

12.7 Exemples de configurations

12.7.1 Supervision continue d'un portail

Contrôles légers planifiés individuellement, toutes les minutes, avec une gigue qui étale les exécutions :

```json
{
  "id": "portail-tcp-443",
  "name": "Portail client : port HTTPS",
  "type": "tcp",
  "schedule": { "interval": "1m", "jitter": "10s" },
  "timeout": "5s",
  "retry": { "attempts": 1, "interval": "15s" },
  "tcp": { "host": "portail.example.com", "port": 443, "tls": true }
}
```

12.7.2 Indicateur métier aux heures d'activité

Contrôle de base de données exécuté tous les quarts d'heure pendant les heures d'activité commerciale, du lundi au samedi (édition Professional, chapitre 11) :

```json
{
  "id": "commandes-du-jour",
  "name": "Volume de commandes de la journée",
  "type": "database",
  "schedule": { "cron": "*/15 9-19 * * 1-6", "timezone": "Europe/Paris", "jitter": "30s" },
  "timeout": "30s",
  "secrets": ["DB_REPORTING_PASSWORD"],
  "database": {
    "engine": "postgresql",
    "host": "db-reporting.example.internal",
    "port": 5432,
    "database": "ventes",
    "username": "supervision_ro",
    "passwordSecret": "DB_REPORTING_PASSWORD",
    "query": "SELECT count(*) AS nb FROM commandes WHERE created_at >= current_date",
    "rules": [
      { "column": "nb", "operator": "gt", "value": 0, "severity": "critical" }
    ],
    "exportValues": [ { "column": "nb", "indicator": "commandes_jour" } ]
  }
}
```

À 9 h 00, le nombre de commandes du jour peut être légitimement faible : la règle se limite donc à vérifier qu'il n'est pas nul. Le chapitre 11 explique comment construire des règles adaptées à l'heure de la journée.

12.7.3 Vérification d'un traitement de nuit

Contrôle exécuté une fois par jour, à 6 h 30, après la fin prévue d'un import nocturne :

```json
{
  "id": "import-nuit-statut",
  "name": "Import nocturne : statut du dernier traitement",
  "type": "database",
  "schedule": { "cron": "30 6 * * *", "timezone": "Europe/Paris" },
  "timeout": "60s",
  "secrets": ["DB_ERP_PASSWORD"],
  "database": {
    "engine": "sqlserver",
    "host": "erp-db.example.internal",
    "port": 1433,
    "database": "erp",
    "username": "supervision_ro",
    "passwordSecret": "DB_ERP_PASSWORD",
    "query": "SELECT TOP 1 statut FROM journal_imports ORDER BY date_fin DESC",
    "rules": [
      { "column": "statut", "operator": "eq", "value": "TERMINE", "severity": "critical", "scope":
   "firstRow" }
    ]
  }
}
```

> **Attention** — Un contrôle exécuté une seule fois par jour devient `STALE` s'il ne produit aucun résultat pendant deux jours. Une défaillance de ce contrôle n'est donc signalée par l'état `STALE` qu'avec un retard important. Pour les traitements critiques, ajoutez un second contrôle plus tardif (par exemple à 8 h), ou surveillez les exécutions en échec dans votre plateforme d'observabilité (chapitre 18).

12.7.4 Parcours de recette en heures ouvrées

Le plan `parcours-boutique` présenté au § 12.3.2 enchaîne quatre scénarios navigateur exécutés uniquement par le plan. Le contrôle de connexion correspondant, sans planification propre, est le suivant :

```json
{
  "id": "boutique-connexion",
  "name": "Boutique : connexion du compte de test (recette)",
  "type": "browser",
  "schedule": null,
  "timeout": "60s",
  "retry": { "attempts": 1, "interval": "30s" },
  "thresholds": { "warningDuration": "20s" },
  "secrets": ["BOUTIQUE_USER", "BOUTIQUE_PASSWORD"],
  "browser": {
    "mode": "steps",
    "steps": [
      { "name": "Ouvrir la page de connexion", "action": "goto", "url": "${param:baseUrl}/connexion"
    },
      { "name": "Saisir l'identifiant", "action": "fill", "selector": "[data-testid=login-email]",
    "value": "${secret:BOUTIQUE_USER}" },
      { "name": "Saisir le mot de passe", "action": "fill", "selector": "[data-testid=login-
    password]", "value": "${secret:BOUTIQUE_PASSWORD}" },
      { "name": "Valider", "action": "click", "selector": "[data-testid=login-submit]",
    "waitForNavigation": true },
      { "name": "Vérifier l'espace client", "action": "assertVisible", "selector": "[data-
    testid=account-menu]" }
    ],
    "params": { "baseUrl": "https://recette.boutique.example.com" },
    "captureOnFailure": true
  }
}
```

La construction des scénarios déclaratifs et scriptés est détaillée au chapitre 14.

12.7.5 Plan de vérification matinale

Plan exécuté une fois par jour ouvré à 7 h 45, avant l'arrivée des utilisateurs, qui vérifie dans l'ordre les briques d'une application et s'arrête au premier échec :

```json
{
  "id": "verification-matinale-erp",
  "name": "ERP : vérification avant ouverture",
  "schedule": { "cron": "45 7 * * 1-5", "timezone": "Europe/Paris" },
  "stopOnFailure": true,
  "maxDuration": "10m",
  "items": [
    { "checkId": "erp-dns" },
    { "checkId": "erp-tcp-443" },
    { "checkId": "erp-portail-http" },
    { "checkId": "import-nuit-statut" },
    { "checkId": "erp-connexion" }
  ]
}
```

Ce plan s'exécute en complément de la supervision continue. En cas d'échec, l'équipe d'exploitation dispose d'un quart d'heure pour intervenir avant l'ouverture. L'ordre des contrôles (DNS, port, page, données, connexion) situe immédiatement le niveau de la panne (chapitre 8, § 8.13).

12.8 Interventions planifiées et fenêtres de maintenance

Pendant une intervention planifiée sur une application supervisée (mise à jour, maintenance de base de données), les contrôles concernés échouent légitimement. Pour éviter des alertes inutiles :

1. avant l'intervention, désactivez les contrôles concernés (chapitre 9, § 9.6.4) et les plans concernés (`POST /api/v1/plans/{id}/disable`, § 12.5.1) ;
2. après l'intervention, réactivez-les, puis lancez une exécution immédiate pour confirmer le retour à la normale ;
3. si vous utilisez une plateforme d'observabilité, déclarez-y aussi une période de maintenance pour suspendre les alertes. Les données remontées pendant l'intervention restent ainsi disponibles pour analyse.

> ✓ **Conseil** — Intégrez ces étapes à votre procédure de gestion des changements. Avec l'API, elles s'automatisent facilement en début et en fin d'intervention (chapitre 9, § 9.13).

> → **Évolution prévue** — Des **fenêtres de maintenance** natives, qui suspendront automatiquement un ensemble de contrôles et de plans sur une période définie à l'avance, sont prévues dans une version ultérieure. Elles éviteront le risque d'oublier de réactiver un contrôle après une intervention.

Ces fenêtres ne doivent pas être confondues avec les **fenêtres de maintenance des mises à jour** de la sonde (section `updates`, paramètre `windows`, chapitre 20). Une fenêtre de mise à jour ne suspend aucun contrôle : elle fixe seulement les créneaux où la sonde peut **activer** une mise à jour, c'est-à-dire redémarrer sur la nouvelle version. L'activation, elle, draine les exécutions en cours puis interrompt la supervision le temps du redémarrage (chapitre 8, § 8.14.2).

> ✓ **Conseil** — Placez les fenêtres de mise à jour en dehors des créneaux où vos plans d'exécution sont les plus importants, et tenez compte de la durée maximale de vos plans au regard du délai de drainage `updates.drainTimeout`.

PARTIE IV

Scénarios de navigation avec Puppeteer

Chapitre 13 — Introduction à Puppeteer

Chapitre 14 — Construction d'un scénario

Chapitre 15 — Exemples progressifs

Chapitre 16 — Bonnes pratiques des scénarios

Chapitre 13 — Introduction à Puppeteer

Ce chapitre ouvre la partie consacrée aux scénarios navigateur. Il présente les technologies employées par la sonde pour simuler un utilisateur : Puppeteer, Chromium et le mode headless. Il explique ce qu'un navigateur automatisé apporte par rapport à une simple requête HTTP, dans quels cas l'utiliser, et quelles sont ses limites. Il s'adresse à tous les lecteurs. Les chapitres suivants détaillent ensuite la construction des scénarios (chapitre 14), des exemples progressifs (chapitre 15) et les bonnes pratiques (chapitre 16).

> **Information** — Les scénarios navigateur (contrôles de type `browser`) sont disponibles dans l'édition **Enterprise** (§ 13.8).

13.1 Présentation de Puppeteer

Puppeteer est une bibliothèque JavaScript, publiée en logiciel libre par l'équipe Chrome de Google, qui pilote un navigateur Chrome ou Chromium par programme. Elle permet d'ouvrir une page, de cliquer sur un bouton, de remplir un formulaire, d'attendre l'apparition d'un élément, de lire le contenu affiché ou de capturer l'écran, exactement comme le ferait un utilisateur, mais de façon automatique et reproductible.

Puppeteer s'exécute dans **Node.js**, l'environnement d'exécution JavaScript côté serveur. Un scénario Puppeteer est donc un programme JavaScript qui décrit, étape par étape, le parcours d'un utilisateur.

L'extrait suivant donne une idée de l'écriture. Il ouvre une page, saisit un terme de recherche, valide et attend les résultats :

JAVASCRIPT

```
// Extrait illustratif : ouverture d'une page et recherche
await page.goto('https://boutique.example.com/recherche');
await page.locator('[data-testid=search-input]').fill('câble');
await page.locator('[data-testid=search-submit]').click();
await page.waitForSelector('[data-testid=search-result]');
```

Chaque instruction est précédée du mot-clé `await`. Les actions sur un navigateur prennent du temps (chargement, réponse du serveur, affichage) : `await` indique au programme d'attendre la fin de l'action avant de passer à la suivante. Le scénario se lit ainsi de haut en bas, dans l'ordre où l'utilisateur agit.

Dans la sonde, Puppeteer est intégré au **moteur de scénarios**. Ce composant isolé lance le navigateur, exécute vos scénarios, mesure chaque étape et produit le résultat (chapitre 3). Vous n'installez rien sur la sonde : vous écrivez le scénario, et la sonde se charge de l'exécuter.

13.2 Chromium et le protocole Chrome DevTools

13.2.1 Rôle de Chromium

Chromium est le projet de navigateur libre sur lequel reposent Google Chrome, Microsoft Edge et de nombreux autres navigateurs. Il comprend le moteur de rendu (Blink), qui interprète le HTML et les feuilles de style, et le moteur JavaScript (V8), qui exécute les scripts des pages.

Utiliser Chromium pour la supervision présente un intérêt majeur : la page est traitée **par le même moteur que celui de la majorité de vos utilisateurs**. Un dysfonctionnement provoqué par un script de la page, une ressource manquante ou une incompatibilité est donc observé dans des conditions proches de la réalité.

La sonde embarque une version de Chromium **fournie par l'éditeur**, associée à la version de Puppeteer correspondante. Vous n'avez pas à l'installer ni à la mettre à jour vous-même : elle fait partie de l'image système signée de la sonde et évolue avec les mises à jour du produit, toujours avec la version de Puppeteer correspondante. Les correctifs de sécurité de Chromium sont publiés sur le canal `security`, accessible sans contrat de maintenance pendant la période de support ; l'éditeur se fixe pour objectifs, non contractuels, de les publier sous 14 jours pour une vulnérabilité critique ou activement exploitée et sous 30 jours pour une vulnérabilité élevée (chapitre 20, § 20.4).

13.2.2 Le protocole Chrome DevTools

Puppeteer ne simule pas les clics par des mouvements de souris à l'écran. Il dialogue avec le navigateur au moyen du **protocole Chrome DevTools** (CDP, *Chrome DevTools Protocol*), l'interface de pilotage qu'utilisent aussi les outils de développement intégrés à Chrome. Par ce canal, Puppeteer peut :

- ouvrir des pages et suivre leurs navigations ;
- interroger le document (DOM, *Document Object Model*, représentation en arbre de la page) ;
- envoyer des événements de souris et de clavier ;
- observer les requêtes réseau, leurs réponses et leurs durées ;
- recevoir les messages de la console JavaScript et les erreurs de la page ;
- capturer l'écran.

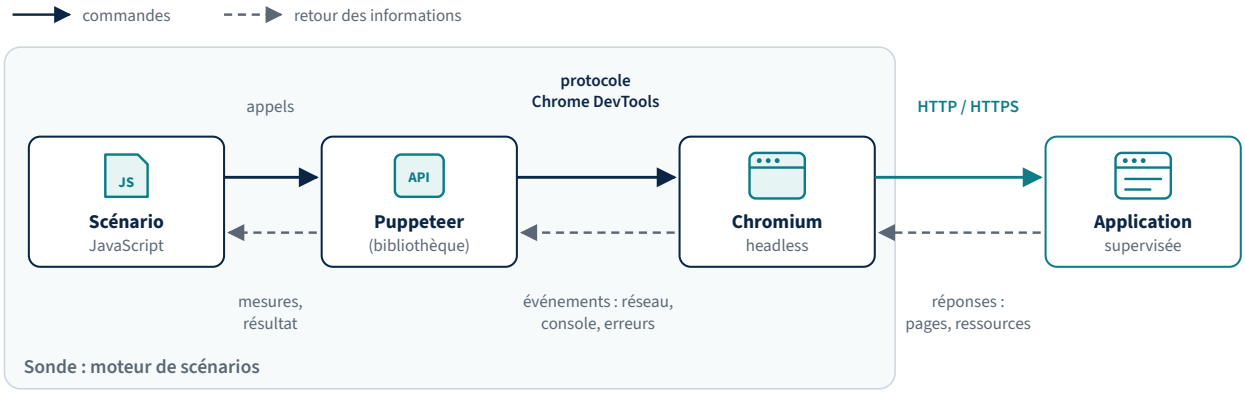


Figure 13-1 — Chaîne d'exécution d'un scénario : scénario, Puppeteer, protocole DevTools, Chromium, application supervisée

Cette architecture explique la richesse des informations disponibles en cas d'échec. La sonde ne sait pas seulement qu'une page n'a pas abouti : elle sait quelle étape a échoué, quelle requête a répondu en erreur et quel message la page a écrit dans sa console.

13.3 Le navigateur headless

13.3.1 Principe

Un navigateur **headless** (« sans tête ») fonctionne normalement, mais sans afficher de fenêtre. Il télécharge les pages, exécute leurs scripts, calcule leur mise en page et produit le même document qu'un navigateur classique ; simplement, rien n'est affiché à l'écran.

Ce mode convient à la sonde, qui ne possède ni écran ni interface graphique. Il consomme aussi moins de ressources qu'un navigateur affiché, puisqu'aucun environnement graphique n'est nécessaire.

Une page rendue en mode headless peut néanmoins faire l'objet d'une capture d'écran. C'est ainsi que la sonde illustre l'étape en échec d'un scénario (chapitre 17).

13.3.2 Mode headless et « headless shell »

Chromium propose deux variantes du fonctionnement headless :

Variante	Description	Usage
Mode headless du navigateur	Le navigateur complet, sans fenêtre. Comportement identique au navigateur affiché	Fidélité maximale au comportement des utilisateurs
Headless shell	Binaire allégé, spécialisé dans l'exécution sans fenêtre	Consommation réduite ; quelques différences de comportement possibles avec le navigateur complet

La sonde utilise le Chromium des dépôts Raspberry Pi OS, dans le **mode headless du navigateur complet** (« new headless »). Ce choix découle de la plateforme : lors de la rédaction de cette documentation, la documentation de Puppeteer sur la configuration système requise (https://pptr.dev/guides/system-requirements) ne mentionne pas *Chrome for Testing* ni le *headless shell* (`chrome-headless-shell`) pour Linux ARM64 ; la documentation de la version en développement (branche `next`) annonce *Chrome for Testing* pour certains systèmes Linux ARM64, la disponibilité réelle dépendant des artefacts publiés pour la version de Puppeteer et la distribution retenues, et Google distribue désormais une version stable de Chrome pour Linux ARM64 (https://blog.google/chromium/bringing-chrome-to-arm64-linux-devices/). Le choix du Chromium des dépôts Raspberry Pi OS est maintenu ; il est réexaminé pour chaque version épinglée de Puppeteer. Il n'a pas d'incidence sur l'écriture des scénarios.

13.3.3 Profil vierge à chaque exécution

À chaque exécution d'un scénario, le moteur de scénarios démarre un navigateur neuf avec un **profil vierge** : aucun cookie, aucune session, aucun cache, aucun historique. Le profil est conservé en mémoire, puis détruit à la fin de l'exécution.

Ce comportement a trois conséquences :

- chaque exécution reproduit la **première visite** d'un utilisateur. Un scénario qui a besoin d'être authentifié doit donc s'authentifier lui-même (chapitre 14, § 14.9) ;
- une exécution ne peut pas être perturbée par l'état laissé par la précédente (panier rempli, session expirée) ;
- aucune donnée de navigation, et donc aucun cookie de session d'un compte de test, ne subsiste sur la carte microSD.

13.4 Navigation automatisée et simple requête HTTP

La sonde propose aussi des contrôles `http` (chapitre 10). Ceux-ci envoient une requête et analysent la réponse. Il est important de comprendre ce qui distingue les deux approches, pour choisir la bonne.

13.4.1 Ce que fait une requête HTTP

Un contrôle `http` envoie **une** requête vers **une** adresse. Il vérifie le code de réponse, les en-têtes, éventuellement le contenu du corps, et mesure les temps de résolution DNS, de connexion, de négociation TLS et de réponse. Il ne charge pas les ressources de la page (images, feuilles de style, scripts) et n'exécute aucun JavaScript.

13.4.2 Ce que fait un navigateur

Un navigateur charge le document, puis **toutes** les ressources qu'il référence : feuilles de style, scripts, images, polices, appels d'API effectués par la page, contenus de services tiers. Il exécute les scripts, qui peuvent eux-mêmes modifier la page, déclencher d'autres requêtes ou rediriger l'utilisateur. Il gère les cookies, les redirections d'authentification et les formulaires.

13.4.3 Comparaison

Aspect	Contrôle `http`	Scénario navigateur
Nombre de requêtes	Une	Toutes celles de la page (souvent plusieurs dizaines)
JavaScript de la page	Non exécuté	Exécuté
Contenu généré dynamiquement	Invisible si produit par JavaScript	Visible
Interactions (clic, saisie)	Impossibles	Possibles
Parcours en plusieurs pages	Limité (contrôle `api` pour les API)	Natif
Services tiers (paiement, cartographie, authentification externe)	Non vérifiés	Vérifiés
Temps mesuré	Temps de réponse du serveur	Temps perçu par l'utilisateur, par étape
Consommation sur la sonde	Très faible	Élevée (un navigateur complet)
Fréquence raisonnable	Toutes les minutes	Toutes les 5 à 15 minutes
Sensibilité aux évolutions de l'interface	Faible	Réelle : les sélecteurs doivent être entretenus

13.4.4 Exemple : une page qui répond, mais qui ne fonctionne pas

Considérons une application dont la page de recherche est servie avec un code HTTP 200. Le formulaire de recherche appelle une API depuis le navigateur pour afficher les résultats. Si cette API est en panne, ou si le script de la page contient une erreur après une mise en production :

- le contrôle `http` sur la page de recherche reste `OK` : la page elle-même est bien servie ;
- le scénario navigateur échoue à l'étape « Vérifier les résultats », car aucun résultat n'apparaît, et il précise si l'API a répondu en erreur.

Les deux contrôles sont complémentaires. Le contrôle `http`, léger, détecte rapidement une indisponibilité franche. Le scénario, plus coûteux, détecte les défaillances du service rendu.

> ✓ **Conseil** — Pour une application critique, associez un contrôle `http` fréquent (toutes les minutes) sur la page d'accueil et un scénario navigateur moins fréquent (toutes les 5 à 10 minutes) sur le parcours principal. Vous obtenez une détection rapide des pannes franches et une vérification approfondie du service.

13.5 Cas d'utilisation en supervision

Les scénarios navigateur répondent aux situations dans lesquelles « le serveur répond » ne suffit pas à affirmer que « le service fonctionne » (chapitre 2).

Cas d'usage	Ce que vérifie le scénario	Exemple du chapitre 15
Disponibilité perçue d'une page	La page se charge et affiche son contenu utile, pas seulement un code 200	Exemples 1 et 2
Authentification	Le compte de test peut se connecter et accède à son espace	Exemple 3
Refus d'authentification	Des identifiants invalides sont refusés avec le message attendu	Exemple 3 bis
Navigation	Les pages principales s'enchaînent, y compris les contenus intégrés (iframes)	Exemple 4
Recherche	La recherche renvoie des résultats pertinents dans un délai acceptable	Exemple 5
Parcours transactionnel	Un produit peut être ajouté au panier	Exemple 6
Tunnel de commande	Le tunnel est franchissable jusqu'à la confirmation, en environnement de test	Exemple 7
Cohérence fonctionnelle	Le résultat affiché est correct (quantité, total), au-delà du message de succès	Exemple 8
Performance perçue	Le temps de chaque étape reste dans les seuils définis	Tous (seuils de durée, chapitre 8)
Services tiers	Le module de paiement, la cartographie ou le fournisseur d'identité répondent	Variantes des exemples 3 et 7

Le scénario mesure la durée de chaque étape. Il permet donc de répondre à trois questions : le parcours a-t-il échoué ? À quelle étape ? Pourquoi ? Les résultats sont détaillés au chapitre 17 et peuvent être transmis aux plateformes d'observabilité (chapitre 18).

13.6 Avantages et limites

13.6.1 Avantages

- **Fidélité.** Le scénario exerce l'application comme un utilisateur, avec le même moteur de navigateur que la plupart d'entre eux.
- **Détection des défaillances fonctionnelles.** Bouton inopérant, script en erreur, API appelée par la page en panne, service tiers défaillant : autant de pannes invisibles pour les contrôles techniques.
- **Diagnostic précis.** Étape en échec, type d'erreur, message, capture d'écran, erreurs de la console et requêtes en erreur.
- **Mesure du temps perçu.** Durée totale et durée par étape.
- **Proactivité.** Le scénario s'exécute en permanence, y compris la nuit et le week-end, lorsque peu d'utilisateurs sont connectés pour signaler un problème.

13.6.2 Limites

Limite	Explication	Conduite à tenir
Consommation de ressources	Un navigateur complet consomme beaucoup de mémoire et de processeur. Sur un Raspberry Pi 5, la sonde n'exécute qu'**un scénario à la fois** par défaut	Espacer les exécutions (5 minutes ou plus) ; regrouper les parcours dans des plans d'exécution (chapitre 12) ; bloquer les ressources tierces inutiles (chapitre 16)
Authentification forte (MFA)	Un second facteur (code SMS, application d'authentification, clé physique) ne peut pas être fourni par un scénario	Prévoir avec les équipes sécurité un compte de test exempté de second facteur, limité en droits et en adresses d'origine
CAPTCHA et protections anti-robots	Ces dispositifs sont conçus pour bloquer les navigateurs automatisés	Demander une exemption pour l'adresse IP de la sonde ou pour le compte de test ; ne jamais chercher à contourner ces protections
Fragilité face aux évolutions de l'interface	Un sélecteur qui ne correspond plus à la page fait échouer le scénario, même si l'application fonctionne	Sélecteurs robustes (`data-testid`), coordination avec les équipes de développement (chapitre 16)
Effets de bord	Un scénario agit réellement sur l'application : commandes, messages, comptes verrouillés	Environnement de test, données de test, garde-fous et nettoyage (exemple 7, chapitre 15)
Volume de trafic	Chaque exécution charge toutes les ressources de la page et peut fausser les statistiques d'audience	Fréquence raisonnable ; identification du trafic de la sonde (chapitre 16, § 16.12)
Représentativité	La sonde mesure le service depuis son propre emplacement réseau, avec son propre matériel	Interpréter les temps comme une tendance ; comparer plusieurs sondes si nécessaire

> ! **Attention** — Un scénario navigateur n'est pas un test de charge. Il vérifie qu'un parcours fonctionne et mesure son temps de réponse. Il ne doit pas être planifié à haute fréquence pour « solliciter » une application.

13.7 De CasperJS à Puppeteer

Cette section s'adresse aux clients qui utilisaient la génération précédente de la solution, dont les scénarios web reposaient sur **CasperJS** et **PhantomJS**. Ces deux outils ne sont plus maintenus depuis plusieurs années, et PhantomJS repose sur un moteur WebKit figé, incapable d'exécuter correctement les applications web modernes. La nouvelle sonde n'exécute donc pas les scénarios CasperJS. Ceux-ci doivent être **réécrits**.

13.7.1 Ce qui ne change pas

Les **intentions** de vos scénarios historiques restent valables, de même que les principes de rédaction que vous appliquiez :

- simuler fidèlement l'utilisateur, par exemple en cliquant dans un champ avant d'y saisir du texte lorsque l'application réagit à ce clic ;
- attendre l'apparition d'un élément plutôt qu'imposer une pause fixe ;
- vérifier un résultat explicite (message, élément, texte) à chaque étape importante ;
- maintenir les scénarios à jour à chaque évolution de l'application.

Avant de réécrire un scénario, formalisez son intention : ce qu'il vérifie, le résultat attendu, le compte de test utilisé. Cette fiche sert ensuite à vérifier que le nouveau scénario détecte les mêmes anomalies.

13.7.2 Ce qui change

CasperJS empilait des étapes dans une file interne (`then`, `waitFor…`) et utilisait des fonctions de rappel de succès et d'échec. Puppeteer s'écrit de façon linéaire, avec `async` et `await` : chaque instruction attend la fin de la précédente. Les échecs sont des **exceptions**, que le moteur de scénarios intercepte pour marquer l'étape en échec. La gestion des erreurs devient ainsi beaucoup plus simple.

La sonde apporte en outre des changements d'organisation :

- le navigateur est lancé et fermé par le moteur de scénarios, pas par le script ;
- les captures d'écran sont réalisées automatiquement en cas d'échec ;
- les identifiants sont fournis par la sonde (secrets), jamais écrits dans le script ;
- les parcours simples peuvent être décrits sans code, en mode déclaratif.

13.7.3 Correspondance des concepts

Pratique CasperJS / PhantomJS	Équivalent avec la sonde et Puppeteer	Remarque
`casper.start(url)`, `casper.thenOpen(url)`	`await page.goto(url, { waitUntil: 'domcontentloaded' })`	Voir chapitre 14, § 14.6
`casper.waitForSelector(sélecteur, succès, échec)`	`await page.waitForSelector(sélecteur, { visible: true })`, ou action sur un *locator*, qui attend automatiquement	Plus de fonctions de rappel : un échec lève une exception
`test.assertExists(sélecteur)`	`await expect.visible(page, sélecteur)`	Assertion fournie par le moteur de scénarios (chapitre 14, § 14.11)
`this.click(sélecteur)`	`await page.locator(sélecteur).click()`	Le *locator* attend que l'élément soit visible et cliquable
`this.sendKeys(sélecteur, texte)`	`await page.locator(sélecteur).fill(texte)`	Le champ est vidé puis rempli
`casper.withFrame(id, fonction)`	Obtenir l'objet `Frame` de l'iframe, puis agir sur ce cadre	Mode déclaratif : propriété `frame` de l'étape (chapitre 14, § 14.13)
`thenEvaluate` + `jQuery.Event('keyup')`	`await page.keyboard.press('Enter')`	Événements clavier natifs : la page n'a plus besoin d'embarquer jQuery
`this.capture(...)` dans le script	Supprimé	Capture automatique par le moteur en cas d'échec
`casper.wait(durée)`	À proscrire	Attendre un élément, une réponse ou une condition (chapitre 14, § 14.10)
Sélecteurs XPath absolus (`/html/body/table[1]/…`)	Attributs dédiés (`data-testid`), rôle d'accessibilité, texte ; XPath en dernier recours	Voir chapitre 14, § 14.7 et chapitre 16
Extension Resurrectio (enregistrement d'actions)	Enregistreur des outils de développement de Chrome	Aide à la conception (chapitre 14, § 14.4)

Pratique CasperJS / PhantomJS	Équivalent avec la sonde et Puppeteer	Remarque
Outil `scenarioRunner` sur poste de travail	Lanceur local fourni avec les exemples, puis exécution d'essai sur la sonde	Chapitre 14, § 14.15 et § 14.16
Identifiants écrits dans le script	Secrets déclarés par le contrôle	Chapitre 9 et chapitre 14, § 14.9

13.7.4 Exemple de réécriture

Le guide historique décrivait un scénario d'authentification qui saisissait volontairement des identifiants invalides, puis vérifiait l'affichage du message « Erreur d'authentification ». Le code CasperJS ressemblait à ceci (extrait reconstitué, sélecteurs simplifiés) :

```javascript
// Ancien style CasperJS — NE FONCTIONNE PAS sur la nouvelle sonde
casper.waitForSelector("input[name='username']",
  function success() {
    this.sendKeys("input[name='username']", "…");
  },
  function fail() {
    test.assertExists("input[name='username']");
  });
```

Le même parcours, réécrit pour la sonde, devient (version complète : exemple 3 bis du chapitre 15) :

```javascript
// Nouveau style : linéaire, attentes intégrées, exceptions en cas d'échec
await step('Soumettre des identifiants invalides', async () => {
  await page.locator('[data-testid=login-email]').fill('compte-inexistant@example.com');
  await page.locator('[data-testid=login-password]').fill('mot-de-passe-volontairement-faux');
  await Promise.all([
    page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
    page.locator('[data-testid=login-submit]').click(),
  ]);
});

await step("Vérifier le message d'erreur", async () => {
  await expect.text(page, '[data-testid=login-error]', "Erreur d'authentification");
});
```

La nouvelle version est plus courte, se lit dans l'ordre du parcours, et ses étapes nommées apparaissent dans les résultats de la sonde.

> **Information** — Il n'existe pas de convertisseur automatique fiable de CasperJS vers Puppeteer : les modèles d'exécution sont trop différents. La réécriture, guidée par la fiche d'intention de chaque scénario, est la démarche recommandée. Pendant la transition, faites fonctionner l'ancien et le nouveau scénario en parallèle et comparez leurs résultats avant de retirer l'ancien.

13.8 Prérequis techniques dans la sonde

13.8.1 Édition et matériel

Prérequis	Valeur
Édition	**Enterprise** (les scénarios déclaratifs et scriptés font partie de ce module)

Prérequis	Valeur
Matériel	Raspberry Pi 5, **8 Go de mémoire vive minimum** (chapitre 4)
Refroidissement	Refroidisseur actif ou boîtier ventilé obligatoire : l'exécution de Chromium sollicite fortement le processeur
Stockage	Carte microSD fournie, suffisante pour un usage standard ; un SSD NVMe facultatif est utile pour des parcours fréquents, comme volume de données (chapitre 20, § 20.3.4)

13.8.2 Composants logiciels embarqués

Composant	Version	Remarque
Node.js	Version LTS, indiquée dans les notes de version et par `GET /api/v1/system` (champ `versions`)	Exécute les scénarios scriptés
Puppeteer (`puppeteer-core`)	Indiquée dans les notes de version et par `GET /api/v1/system` (champ `versions`)	Version publiée par la sonde dans `GET /api/v1/system`
Chromium	Indiquée dans les notes de version et par `GET /api/v1/system` (champ `versions`)	Issu des dépôts Raspberry Pi OS, retenu par l'éditeur et intégré à l'image système signée ; mis à jour avec le produit (§ 13.3.2, chapitre 20)
Modules JavaScript disponibles pour les scripts	Aucun	Un script ne peut ni importer ni installer de module : seuls les objets du contrat d'exécution sont disponibles (chapitre 14, § 14.5.1)

`puppeteer-core` est la variante de Puppeteer qui ne télécharge pas de navigateur et utilise celui qui lui est fourni. Elle est nécessaire sur le Raspberry Pi 5, dont l'architecture ARM 64 bits n'est pas couverte par les navigateurs que Puppeteer télécharge habituellement : lors de la rédaction de cette documentation, la documentation de Puppeteer ne liste pas *Chrome for Testing* pour Linux ARM64 (§ 13.3.2). La disponibilité effective des artefacts peut varier selon le canal et la version ; elle est vérifiée pour chaque version épinglée , et le choix du Chromium des dépôts Raspberry Pi OS reste la référence tant que cette vérification n'a pas conclu autrement. Cette particularité est transparente pour l'auteur de scénarios : l'API utilisée dans les scripts est la même.

> **Information** — Les exemples de cette documentation ont été écrits et testés sur poste de travail avec **Puppeteer 25.18.1** (Chrome for Testing 157), au moyen du lanceur local fourni avec les exemples. Ils n'utilisent que des fonctions stables de l'API Puppeteer : *locators*, `waitForSelector`, `waitForResponse`, `waitForFunction`, `waitForNavigation`, `keyboard`, et les sélecteurs `::-p-text()`, `::-p-aria()` et `::-p-xpath()`. Avant de déployer un scénario, vérifiez la version de Puppeteer de votre sonde dans `GET /api/v1/system` et effectuez une exécution d'essai (chapitre 14, § 14.16).

13.8.3 Capacité d'exécution

Le moteur de scénarios exécute par défaut **un seul scénario à la fois**. Sur une carte de 8 Go, cette limite peut être portée à deux après vérification de la consommation réelle de vos scénarios. Les scénarios dont l'échéance arrive pendant une exécution attendent leur tour. Le retard est mesuré et visible dans la supervision interne de la sonde (état de santé au chapitre 17, métriques de santé au chapitre 18).

Pour estimer la capacité nécessaire, multipliez la durée moyenne de chaque scénario par sa fréquence d'exécution horaire. Le total doit rester nettement inférieur à 60 minutes par heure et par emplacement d'exécution :

Scénario	Durée moyenne	Fréquence	Occupation par heure
Connexion	8 s	toutes les 5 min (12/h)	96 s

Scénario	Durée moyenne	Fréquence	Occupation par heure
Recherche	6 s	toutes les 5 min (12/h)	72 s
Tunnel de commande (test)	25 s	toutes les 15 min (4/h)	100 s
Total			**268 s, soit environ 4,5 min/h**

Dans cet exemple fictif, l'occupation reste faible et laisse une large marge pour les nouvelles tentatives et les exécutions manuelles. Le chapitre 12 détaille la planification et le regroupement des scénarios en plans d'exécution.

13.8.4 Prérequis réseau

Le navigateur de la sonde doit pouvoir joindre **toutes** les ressources des pages parcourues, et pas seulement l'adresse principale de l'application : réseaux de diffusion de contenu, services d'authentification, modules de paiement de test, API appelées par la page. Recensez ces destinations avec les équipes de l'application et autorisez-les dans le pare-feu et, le cas échéant, dans le proxy (chapitre 7).

Une ressource bloquée par le pare-feu peut faire échouer un scénario, alors que l'application fonctionne pour les utilisateurs. À l'inverse, les ressources inutiles à la vérification (mesure d'audience, publicité) peuvent être volontairement bloquées par le scénario (chapitre 16, § 16.11).

13.8.5 Poste de développement

Pour écrire et mettre au point les scénarios, un poste de travail équipé de Node.js et de Puppeteer est recommandé. Le dossier d'exemples `exemples/puppeteer/`, livré dans l'archive `synovex-exemples.zip` avec la documentation, contient une application de démonstration fictive et un **lanceur local** qui reproduit le contrat d'exécution de la sonde. Leur utilisation est décrite au chapitre 14, § 14.15.

Chapitre 14 — Construction d'un scénario

Ce chapitre explique comment construire un scénario navigateur, de sa conception à son déploiement sur la sonde. Il présente les deux modes d'écriture proposés par la sonde, le mode déclaratif et le mode scripté. Il détaille ensuite chaque aspect d'un scénario scripté : ouverture des pages, recherche d'éléments, interactions, authentification, attentes, assertions et traitement des erreurs. Il se termine par la mise au point sur poste de travail, le déploiement et le cycle de vie des scénarios.

Les exemples s'appuient sur la **Boutique Démo**, application web **fictive** fournie avec les exemples de la documentation (dossier `exemples/puppeteer/`). Les adresses `boutique.example.com` et le compte `test-supervision@example.com` sont également fictifs.

14.1 Deux modes d'écriture

Un scénario est un contrôle de type `browser` (chapitre 8). Son contenu peut être décrit de deux manières.

	Scénario déclaratif	Scénario scripté
Forme	Liste d'étapes JSON (`browser.mode = "steps"`)	Module JavaScript Puppeteer (`browser.mode = "script"`)
Compétences requises	Aucune programmation ; connaissance des sélecteurs	JavaScript et API Puppeteer
Rôle nécessaire pour le créer ou en changer le code	`operator`	`admin` (le script est du code exécuté sur le réseau où la sonde est installée). Un `operator` ne peut ni définir ou retirer `browser.mode: script`, ni modifier `browser.scriptRef` : la sonde répond `403`. Il peut en revanche activer, désactiver, déclencher, ajuster par `PATCH` la planification ou les seuils, et supprimer le contrôle
Validation	Par schéma, à l'enregistrement : une erreur de structure est refusée immédiatement	À l'exécution d'essai
Logique conditionnelle, boucles, calculs	Non	Oui
Lecture de valeurs et comparaison entre étapes	Non	Oui
Attente d'une réponse d'API précise	Non	Oui
Garde-fous (vérifier l'environnement avant une action)	Limités	Complets
Maintenance	Très simple	Demande un développeur

14.1.1 Choisir le mode

Choisissez le **mode déclaratif** lorsque le parcours est une suite linéaire d'actions et de vérifications : ouvrir une page, cliquer, saisir, vérifier qu'un élément ou un texte apparaît. C'est le cas de la majorité des scénarios de disponibilité et d'authentification.

Choisissez le **mode scripté** lorsque le scénario doit :

- lire une valeur dans la page et la comparer à un résultat calculé (contrôle d'un total, d'une quantité) ;
- attendre la réponse d'une API appelée par la page ;
- adapter son comportement à ce qu'il trouve (fermer un bandeau s'il est présent) ;
- vérifier des conditions de sécurité avant une action irréversible (environnement de test) ;
- manipuler des fenêtres secondaires ou des boîtes de dialogue.

> ✓ **Conseil** — Commencez par un scénario déclaratif. Passez au mode scripté seulement lorsque le besoin l'exige. Un scénario déclaratif est plus simple à relire, à confier à une autre équipe et à maintenir.

14.2 Paramètres d'un contrôle navigateur

Quel que soit le mode, un scénario est un contrôle qui possède les paramètres communs décrits au chapitre 8 (planification, timeout, nouvelles tentatives, seuils, secrets, étiquettes) et un bloc `browser` spécifique.

JSON

```
{
  "id": "boutique-connexion",
  "name": "Boutique : connexion du compte de test",
  "type": "browser",
  "schedule": { "interval": "5m", "jitter": "30s" },
  "timeout": "60s",
  "retry": { "attempts": 1, "interval": "30s" },
  "thresholds": { "warningDuration": "8s" },
  "secrets": ["BOUTIQUE_USER", "BOUTIQUE_PASSWORD"],
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "https://boutique.example.com" },
    "viewport": { "width": 1366, "height": 768 },
    "captureOnFailure": true,
    "blockDomains": ["stats.example.net"],
    "userAgentSuffix": "SupervisionSonde",
    "steps": [
      { "name": "Ouvrir la page de connexion", "action": "goto", "url": "${param:baseUrl}/connexion"
    }
    ]
  }
}
```

Dans cet extrait, le tableau `steps` est abrégé : sa forme complète est présentée au § 14.3.

Propriété de `browser`	Rôle	Valeur par défaut
`mode`	`steps` (déclaratif) ou `script` (scripté)	Obligatoire
`steps`	Étapes du scénario déclaratif (50 au maximum)	Obligatoire en mode `steps`
`scriptRef`	Référence immuable du script téléversé (`nom@sha256:empreinte`)	Obligatoire en mode `script`

Propriété de `browser`	Rôle	Valeur par défaut
`params`	Paramètres non sensibles transmis au scénario (adresse de base, terme de recherche, identifiant de produit)	Aucun
`viewport`	Taille de la fenêtre simulée, en pixels	1366 × 768
`captureOnFailure`	Capture d'écran automatique en cas d'échec	`true`
`blockDomains`	Domaines dont les requêtes sont bloquées (mesure d'audience, publicité)	Aucun
`userAgentSuffix`	Suffixe ajouté à l'identification du navigateur, pour reconnaître le trafic de la sonde	Aucun

Les paramètres `params` rendent un même scénario réutilisable : l'adresse de base change entre la recette et la production, sans modification du scénario lui-même.

> ! **Avertissement** — Ne placez jamais un mot de passe, un jeton ou une donnée personnelle dans `params`. Ces valeurs sont visibles par tout utilisateur pouvant consulter le contrôle. Les informations sensibles sont des **secrets** (§ 14.9 et chapitre 9).

14.3 Scénario déclaratif

14.3.1 Structure

Un scénario déclaratif est une liste ordonnée d'étapes. Chaque étape possède un **nom**, qui apparaît dans les résultats, et une **action**. Les étapes sont exécutées dans l'ordre. La première étape en échec arrête le scénario : le contrôle passe à l'état `CRITICAL` et le résultat indique le nom de cette étape.

```json
"steps": [
  { "name": "Ouvrir la page de connexion", "action": "goto", "url": "${param:baseUrl}/connexion" },
  { "name": "Saisir l'identifiant", "action": "fill", "selector": "[data-testid=login-email]",
    "value": "${secret:BOUTIQUE_USER}" },
  { "name": "Saisir le mot de passe", "action": "fill", "selector": "[data-testid=login-password]",
    "value": "${secret:BOUTIQUE_PASSWORD}" },
  { "name": "Valider le formulaire", "action": "click", "selector": "[data-testid=login-submit]",
    "waitForNavigation": true },
  { "name": "Vérifier l'accès à l'espace client", "action": "assertVisible", "selector": "[data-
    testid=account-menu]" }
]
```

Le navigateur est ouvert avant la première étape et fermé après la dernière, par le moteur de scénarios. Vous n'avez rien à prévoir pour cela.

14.3.2 Propriétés d'une étape

Propriété	Rôle
`name`	Nom affiché dans les résultats et la télémétrie. 80 caractères au maximum. Choisissez un libellé **stable et explicite** (« Valider le formulaire », et non « Étape 4 »)
`action`	Action à réaliser (§ 14.3.3)
`selector`	Sélecteur de l'élément visé (§ 14.7)
`url`	Adresse à ouvrir (`goto`)

Propriété	Rôle
`value`	Valeur saisie, sélectionnée ou attendue
`key`	Touche du clavier (`press`)
`operator`	Mode de comparaison des assertions de texte, d'URL et de titre
`frame`	Sélecteur de l'iframe contenant l'élément (§ 14.13)
`timeout`	Délai maximal de l'étape (`5s` , `15s` …) ; à défaut, délai par défaut du moteur (10 s)
`waitForNavigation`	`true` : attendre le chargement de la nouvelle page provoqué par un clic ou une touche

14.3.3 Référence des actions

Actions de navigation et d'interaction

Action	Propriétés	Effet	Exemple
`goto`	`url`	Ouvre une adresse et attend que le document soit chargé	`{ "name": "Ouvrir l'accueil", "action": "goto", "url": "${param:baseUrl}/" }`
`click`	`selector` , `waitForNavigation`	Attend que l'élément soit visible et cliquable, puis clique	`{ "name": "Ouvrir le catalogue", "action": "click", "selector": "[data-testid=nav-catalogue]", "waitForNavigation": true }`
`fill`	`selector` , `value`	Vide le champ puis saisit la valeur	`{ "name": "Saisir le terme", "action": "fill", "selector": "[data-testid=search-input]", "value": "câble" }`
`select`	`selector` , `value`	Choisit l'option d'une liste déroulante, désignée par sa **valeur technique** (attribut `value` de l'option)	`{ "name": "Choisir le pays", "action": "select", "selector": "#pays", "value": "be" }`
`press`	`key` , `selector` (facultatif), `waitForNavigation`	Appuie sur une touche. Si un sélecteur est fourni, l'élément est d'abord cliqué pour recevoir le focus	`{ "name": "Valider par Entrée", "action": "press", "selector": "[data-testid=search-input]", "key": "Enter" }`
`hover`	`selector`	Survole un élément (menus déroulants)	`{ "name": "Déplier le menu", "action": "hover", "selector": "[data-testid=menu-compte]" }`
`waitForSelector`	`selector`	Attend qu'un élément devienne visible	`{ "name": "Attendre les résultats", "action": "waitForSelector", "selector": "[data-testid=search-result]" }`
`waitForNavigation`	—	Attend le chargement d'une nouvelle page	Voir l'attention ci-dessous

Les noms de touches de `press` sont ceux du clavier standard : `Enter` , `Tab` , `Escape` , `ArrowDown` , `Backspace` , `Digit7` (chiffre 7 de la rangée principale), `KeyA` , etc.

> **Attention** — L'action autonome `waitForNavigation` attend la **prochaine** navigation. Si le chargement de la page a déjà eu lieu au moment où l'étape commence, l'attente échoue au bout du délai. Préférez toujours la propriété `"waitForNavigation": true` sur l'étape `click` ou `press` qui déclenche le changement de page : l'attente est alors armée avant l'action.

Actions de vérification (assertions)

Action	Propriétés	Réussit si…
`assertVisible`	`selector`	L'élément est visible dans le délai imparti
`assertHidden`	`selector`	L'élément est absent ou masqué dans le délai imparti
`assertText`	`selector`, `value`, `operator`	Le texte de l'élément vérifie la comparaison
`assertUrl`	`value`, `operator`	L'adresse courante vérifie la comparaison
`assertTitle`	`value`, `operator`	Le titre de la page vérifie la comparaison

`operator`	Signification	Exemple de `value`
`contains` (par défaut)	Le texte contient la valeur	`"Bienvenue"`
`notContains`	Le texte ne contient pas la valeur	`"Erreur"`
`eq`	Le texte est exactement égal à la valeur (espaces de début et de fin ignorés)	`"1"`
`ne`	Le texte est différent de la valeur	`"0"`
`matches`	Le texte correspond à l'expression régulière	`"^[1-9][0-9]* résultat"`

14.3.4 Variables

Les valeurs `url` et `value` peuvent contenir des références, remplacées au moment de l'exécution :

Syntaxe	Source	Exemple
`${param:nom}`	Paramètre de `browser.params`	`"${param:baseUrl}/connexion"`
`${secret:NOM}`	Secret déclaré dans la liste `secrets` du contrôle	`"${secret:BOUTIQUE_PASSWORD}"`

Une référence vers un paramètre absent, ou vers un secret non déclaré dans `secrets`, est une erreur. Le contrôle est refusé à l'enregistrement, ou l'exécution échoue avec l'état `UNKNOWN`.

> **Conseil** — Utilisez `${secret:…}` uniquement dans les étapes `fill`. Une assertion qui compare un texte de la page à un secret risquerait de faire apparaître ce secret dans un message d'erreur. La sonde masque les valeurs de secrets connues dans les journaux, mais il est préférable de ne pas les exposer à ce risque.

14.3.5 Exemple complet

Le scénario suivant vérifie le moteur de recherche de la Boutique Démo. Il correspond au fichier `exemples/puppeteer/scenarios/05-recherche.json`.

JSON

```
{
  "id": "demo-recherche",
  "name": "Boutique Démo : moteur de recherche",
  "type": "browser",
  "schedule": { "interval": "10m" },
  "timeout": "45s",
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080", "terme": "câble" },
    "steps": [
      { "name": "Ouvrir la page de recherche", "action": "goto", "url": "${param:baseUrl}/recherche"
    },
      { "name": "Saisir le terme recherché", "action": "fill", "selector": "[data-testid=search-
    input]", "value": "${param:terme}" },
      { "name": "Lancer la recherche", "action": "click", "selector": "[data-testid=search-submit]"
    },
      { "name": "Vérifier l'affichage des résultats", "action": "assertVisible", "selector": "[data-
    testid=search-result]" },
      { "name": "Vérifier le compteur de résultats", "action": "assertText", "selector": "[data-
    testid=search-count]", "operator": "matches", "value": "^[1-9][0-9]* résultat" }
    ]
  }
}
```

La recherche de cette application ne charge pas de nouvelle page : les résultats sont affichés par un script. L'étape « Lancer la recherche » n'a donc pas de `waitForNavigation`. L'étape suivante, `assertVisible`, attend l'apparition du premier résultat dans la limite de son délai.

14.3.6 Limites du mode déclaratif

Le mode déclaratif ne permet pas de :

- comparer deux valeurs lues dans la page, ou une valeur et un résultat calculé ;
- répéter une action ou choisir une action selon le contenu de la page ;
- attendre une réponse réseau précise ;
- manipuler une fenêtre secondaire ou une boîte de dialogue.

Lorsque l'un de ces besoins apparaît, convertissez le scénario en mode scripté. La conversion est directe : chaque étape déclarative devient un appel `step()` contenant l'instruction Puppeteer équivalente (§ 14.5).

14.4 Concevoir un scénario avec l'enregistreur de Chrome

Les outils de développement de Google Chrome comprennent un **enregistreur** (panneau *Recorder*) qui consigne les actions réalisées dans le navigateur : navigation, clics, saisies. L'enregistrement peut être exporté au format JSON ou sous forme de script Puppeteer. Cet outil succède, pour cet usage, à l'extension Resurrectio qu'utilisait la génération précédente.

14.4.1 Procédure

Objectif : obtenir rapidement la liste des actions et des sélecteurs d'un parcours.

Prérequis : Google Chrome sur un poste de travail ayant accès à l'application, idéalement en environnement de test.

Étapes

1. Ouvrez l'application dans Chrome, puis les outils de développement (touche F12).
2. Ouvrez le panneau **Recorder** (menu des outils de développement, *More tools*, puis *Recorder*, selon la version de Chrome).
3. Créez un nouvel enregistrement, nommez-le, puis démarrez-le.
4. Réalisez le parcours comme un utilisateur.
5. Arrêtez l'enregistrement.
6. Exportez-le au format **Puppeteer** (script JavaScript) ou **JSON**.

14.4.2 Adapter l'enregistrement

L'enregistrement est un **brouillon**, pas un scénario prêt à déployer. Il doit être retravaillé :

Point à reprendre	Raison
Supprimer le lancement et la fermeture du navigateur	Le moteur de scénarios les prend en charge
Regrouper les actions en étapes nommées (`step()` ou `name`)	Les résultats doivent indiquer l'étape métier en échec
Remplacer les sélecteurs générés par des sélecteurs stables (`data-testid`)	Les sélecteurs générés, positionnels ou dépendants du style, cassent à la moindre évolution (§ 14.7)
Supprimer les actions inutiles (clics de positionnement, défilements)	Elles allongent le scénario et le fragilisent
Remplacer les identifiants saisis par des références à des secrets	Les identifiants ne doivent jamais figurer dans un scénario
Ajouter les **assertions**	L'enregistreur consigne des actions, pas des vérifications. Un scénario sans assertion ne détecte presque rien
Supprimer les dépendances propres aux exports (bibliothèques d'exécution d'enregistrements)	Seul le contrat de la sonde est disponible dans le moteur

> → **Évolution prévue** — L'import direct d'un enregistrement JSON de l'enregistreur de Chrome sous forme de scénario déclaratif est prévu dans une version ultérieure, avec les importateurs du format historique et des projets SoapUI. Aucun importateur n'est proposé : l'enregistrement sert de base à une écriture manuelle.

14.5 Scénario scripté : structure et contrat

14.5.1 Le contrat du module

Un scénario scripté est un **module JavaScript** (extension `.mjs`). Il exporte par défaut une fonction asynchrone et ne peut importer aucun autre module (`import`). Le moteur de scénarios appelle cette fonction en lui transmettant un **objet de contexte**, qui contient tout ce dont le script dispose :

```javascript
export default async function scenario({ page, step, secrets, params, log, expect }) {
  // Le parcours est écrit ici, sous forme d'étapes.
}
```

Élément	Description
`page`	Objet `Page` de Puppeteer. Le navigateur est déjà lancé, avec un profil vierge. Le script **ne lance ni ne ferme** le navigateur
`step(nom, fonction)`	Déclare une étape mesurée. Le nom doit être **littéral et stable** (80 caractères au maximum, 50 étapes au maximum). Renvoie la valeur renvoyée par la fonction
`secrets`	Objet en lecture seule contenant **uniquement** les secrets déclarés par le contrôle
`params`	Paramètres non sensibles du contrôle (`browser.params`)
`log.info(message)` , `log.warn(message)`	Ajoute un message au journal de l'exécution. Les valeurs des secrets sont masquées
`expect`	Fonctions d'assertion (§ 14.11)

14.5.2 Structure type

JAVASCRIPT

```
// Scénario : connexion du compte de test à la boutique (application fictive).
// Paramètres : baseUrl. Secrets : BOUTIQUE_USER, BOUTIQUE_PASSWORD.

export default async function scenario({ page, step, secrets, params, expect }) {
  await step('Ouvrir la page de connexion', async () => {
    await page.goto(`${params.baseUrl}/connexion`, { waitUntil: 'domcontentloaded' });
    await expect.visible(page, '[data-testid=login-form]');
  });

  await step('Saisir les identifiants', async () => {
    await page.locator('[data-testid=login-email]').fill(secrets.BOUTIQUE_USER);
    await page.locator('[data-testid=login-password]').fill(secrets.BOUTIQUE_PASSWORD);
  });

  await step('Valider le formulaire', async () => {
    await Promise.all([
      page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
      page.locator('[data-testid=login-submit]').click(),
    ]);
  });

  await step("Vérifier l'accès à l'espace client", async () => {
    await expect.visible(page, '[data-testid=account-menu]');
    expect.url(page, '/compte');
  });
}
```

Un scénario bien construit présente les caractéristiques suivantes :

- un commentaire d'en-tête indique l'application, les paramètres et les secrets attendus ;
- chaque étape correspond à une **intention métier** compréhensible par un exploitant ;
- chaque étape importante se termine par une **vérification** ;
- aucune valeur sensible et aucune adresse propre à un environnement ne figure dans le code.

14.5.3 Déroulement d'une exécution

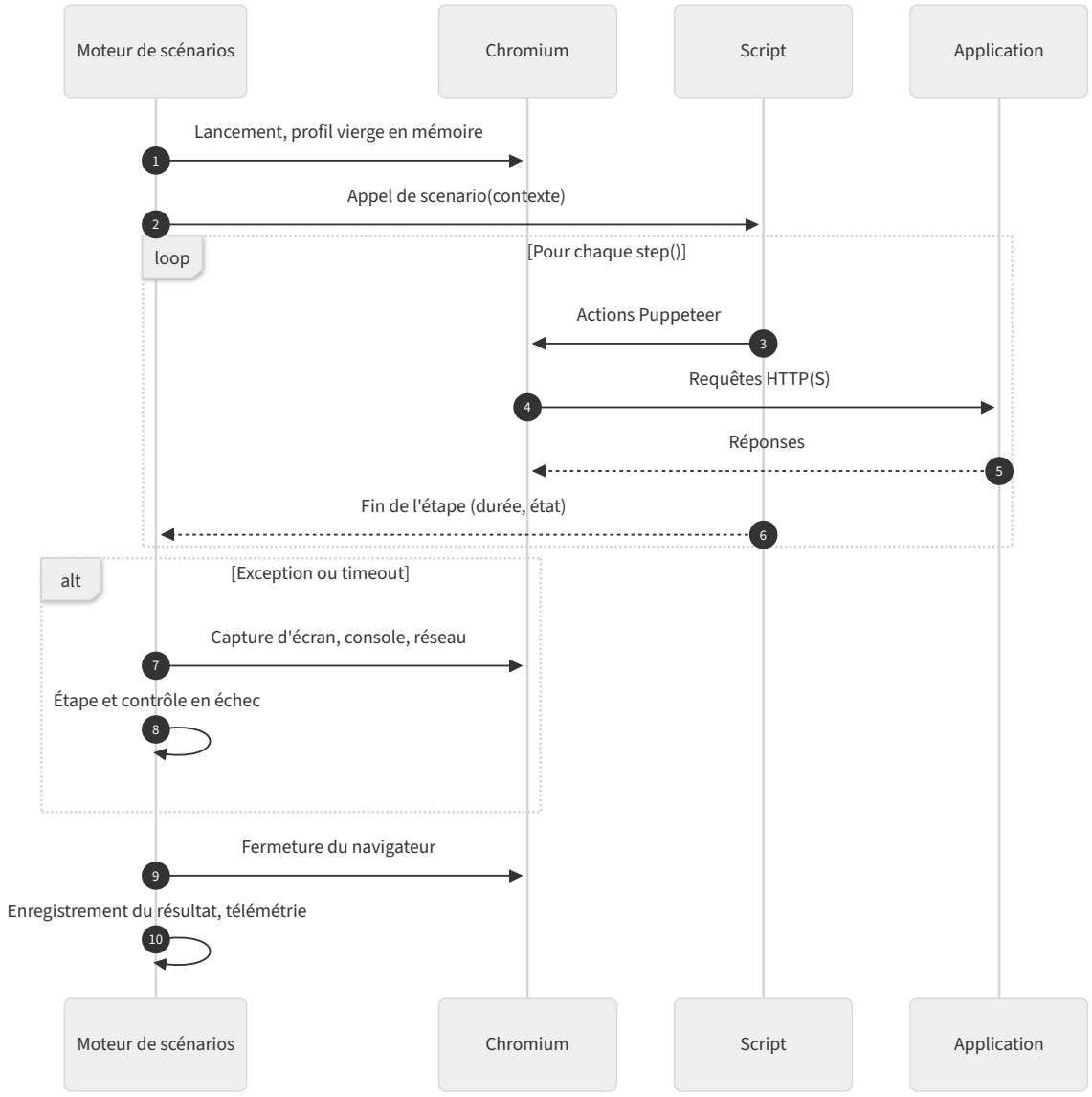


Schéma 14-1 — Déroulement de l'exécution d'un scénario scripté

14.5.4 Le code en dehors des étapes

Le code placé hors de tout `step()` s'exécute normalement, mais n'est pas mesuré. Réservez-le aux préparations sans action sur la page : lecture des paramètres, déclaration de fonctions utilitaires, abonnement à des événements.

```javascript
export default async function scenario({ page, step, params }) {
  // Préparation hors étape : valeurs par défaut des paramètres
  const produit = params.produit || 'P100';

  // Fonction utilitaire réutilisée par plusieurs étapes
  const cliquerEtAttendre = (selecteur) =>
    Promise.all([
      page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
      page.locator(selecteur).click(),
    ]);

  await step('Ouvrir le catalogue', async () => {
    await page.goto(`${params.baseUrl}/`, { waitUntil: 'domcontentloaded' });
    await cliquerEtAttendre('[data-testid=nav-catalogue]');
  });
  // …
}
```

> **Attention** — Le nom d'une étape doit rester **identique d'une exécution à l'autre**. N'y insérez pas de valeur variable (date, quantité, identifiant). Les noms d'étapes servent à agréger les mesures dans le temps et dans les plateformes d'observabilité. Un nom variable créerait une nouvelle série de mesures à chaque exécution.

14.6 Ouvrir une page et naviguer

14.6.1 Ouvrir une adresse

```javascript
const reponse = await page.goto(`${params.baseUrl}/`, { waitUntil: 'domcontentloaded' });
expect.ok(reponse && reponse.ok(), `Code HTTP inattendu : ${reponse?.status()}`);
```

`page.goto()` renvoie la réponse HTTP du document principal, ce qui permet de vérifier son code. L'option `waitUntil` détermine le moment où la navigation est considérée comme terminée :

Valeur	Terminé lorsque…	Usage recommandé
`domcontentloaded`	Le document HTML est analysé, avant le chargement des images	**Valeur recommandée**, à compléter par l'attente de l'élément utile
`load`	La page et toutes ses ressources (images, feuilles de style) sont chargées	Mesure du chargement complet ; sensible aux ressources tierces lentes
`networkidle2`	Au plus deux connexions réseau actives pendant 500 ms	Pages à chargement progressif ; peut ne jamais aboutir sur une page qui interroge régulièrement le serveur
`networkidle0`	Aucune connexion réseau active pendant 500 ms	Déconseillé en supervision : les pages modernes maintiennent souvent des connexions

La pratique la plus robuste consiste à attendre `domcontentloaded`, puis l'élément qui prouve que la page est **utilisable** (formulaire, liste de résultats, menu). On mesure ainsi le temps utile pour l'utilisateur, sans dépendre des ressources secondaires.

14.6.2 Naviguer par un clic

Lorsqu'un clic provoque le chargement d'une nouvelle page, l'attente de la navigation doit être **armée avant le clic**. Sinon, une page très rapide peut être chargée avant que l'attente ne commence, et celle-ci échoue alors au bout du délai.

```javascript
// Bonne pratique : attente et clic lancés ensemble
await Promise.all([
  page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
  page.locator('[data-testid=nav-catalogue]').click(),
]);
```

`Promise.all` lance les deux opérations simultanément et attend qu'elles soient toutes deux terminées.

14.6.3 Pages à navigation interne

Dans une application monopage, construite avec un framework JavaScript, un clic modifie souvent le contenu et l'adresse **sans charger de nouvelle page**. `waitForNavigation` ne convient pas dans ce cas : attendez l'élément caractéristique de la nouvelle vue.

```javascript
await page.locator('[data-testid=nav-commandes]').click();
await page.waitForSelector('[data-testid=orders-table]', { visible: true });
```

14.6.4 Autres opérations de navigation

Opération	Instruction
Revenir à la page précédente	`await page.goBack({ waitUntil: 'domcontentloaded' })`
Recharger la page	`await page.reload({ waitUntil: 'domcontentloaded' })`
Lire l'adresse courante	`page.url()`
Lire le titre	`await page.title()`

14.7 Rechercher des éléments

14.7.1 Les sélecteurs

Un **sélecteur** désigne un ou plusieurs éléments de la page. Puppeteer accepte les sélecteurs CSS standard, ainsi que des sélecteurs étendus propres à Puppeteer, préfixés par `::-p-`.

Type	Exemple	Robustesse	Commentaire
Attribut dédié aux tests	`[data-testid=login-submit]`	**Très bonne**	Recommandé : attribut posé volontairement par les développeurs, indépendant du style et des textes
Identifiant	`#login-submit`	Bonne	Si l'identifiant est stable (pas généré automatiquement)
Rôle et nom d'accessibilité	`::-p-aria(Valider la commande)`	Bonne	Désigne l'élément par son nom accessible (libellé, `aria-label`)
Texte visible	`::-p-text(Ajouter au panier)`	Moyenne	Lisible ; dépend des libellés et de la langue

Type	Exemple	Robustesse	Commentaire
Attribut fonctionnel	`input[name=email]`, `form[action="/connexion"]`	Moyenne	Correct si l'attribut a un sens fonctionnel
Classe CSS	`.btn-primary`	Faible	Les classes servent au style et changent souvent
Position	`div > ul > li:nth-child(3) > a`	**Très faible**	Casse au moindre changement de mise en page
XPath	`::-p-xpath(//button[contains(., "Valider")])`	Variable	En dernier recours ; jamais de chemin absolu du type `/html/body/div[2]/…`

> ✓ **Conseil** — Demandez aux équipes de développement d'ajouter des attributs `data-testid` aux éléments utilisés par vos scénarios (chapitre 16, § 16.16). C'est la mesure qui réduit le plus les faux positifs liés aux évolutions de l'interface.

Les sélecteurs se testent directement dans la console des outils de développement de Chrome. `document.querySelectorAll('[data-testid=search-result]')` renvoie les éléments correspondants. Vérifiez qu'un sélecteur désigne **un seul** élément lorsque vous voulez agir dessus.

14.7.2 Locators et handles

Puppeteer propose deux façons d'accéder aux éléments.

Les *locators* (`page.locator(sélecteur)`) décrivent **comment trouver** un élément. L'élément est recherché au moment de l'action, et l'action attend automatiquement qu'il soit présent, visible, stable (sans animation en cours) et, pour un clic ou une saisie, activé. C'est la méthode recommandée pour agir sur la page.

```javascript
await page.locator('[data-testid=add-to-cart]').click();
await page.locator('[data-testid=quantity]').fill('2');
```

Les *handles* (`page.$()`, `page.$$()`, `page.waitForSelector()`) renvoient une **référence** à un élément trouvé à un instant donné. Ils servent surtout à lire des informations.

```javascript
// Lire le texte d'un élément
const total = await page.$eval('[data-testid=cart-total]', (el) => el.textContent.trim());

// Compter des éléments
const lignes = await page.$$('[data-testid=cart-line]');

// Lire un attribut technique plutôt qu'un texte formaté
const prix = Number(await page.$eval('[data-testid=product-price]', (el) => el.dataset.value));
```

`page.$eval(sélecteur, fonction)` exécute la fonction **dans la page**, sur le premier élément correspondant, et renvoie le résultat au scénario. `page.$$eval` fait de même sur la liste de tous les éléments correspondants. Si aucun élément ne correspond, `page.$eval` lève une erreur. Assurez-vous donc de la présence de l'élément au préalable, par exemple avec `expect.visible`.

> ℹ **Information** — Le code passé à `$eval`, `$$eval` ou `waitForFunction` s'exécute dans le navigateur, et non dans le scénario. Il n'a pas accès aux variables du scénario, sauf si elles sont passées explicitement en argument (§ 14.10.4).

14.8 Interagir avec la page

14.8.1 Cliquer

```javascript
await page.locator('[data-testid=add-to-cart]').click();
```

Le clic est réalisé au centre de l'élément, après son défilement dans la zone visible. Puppeteer clique à cette position : si un autre élément recouvre la cible, par exemple un bandeau de consentement aux cookies, c'est cet élément qui reçoit le clic. Le bouton visé n'est alors pas actionné, et c'est la vérification de l'étape suivante qui échoue, avec un diagnostic moins clair. Fermez donc d'abord le bandeau (§ 14.8.6).

14.8.2 Saisir du texte

```javascript
// fill() vide le champ, puis saisit la valeur
await page.locator('[data-testid=login-email]').fill(secrets.BOUTIQUE_USER);
```

Certains champs réagissent à chaque frappe : autocomplétion, masque de saisie. `fill()` peut alors ne pas déclencher le comportement attendu. Dans ce cas, simulez la frappe touche par touche :

```javascript
await page.locator('[data-testid=search-input]').click();
await page.keyboard.type('câble', { delay: 50 }); // 50 ms entre deux frappes
```

14.8.3 Listes, cases à cocher et boutons radio

```javascript
// Liste déroulante : sélection par la valeur technique de l'option
await page.select('#pays', 'be');

// Case à cocher ou bouton radio : un clic
await page.locator('[data-testid=accept-cgv]').click();
await page.locator('[data-testid=delivery-pickup]').click();

// Vérifier l'état d'une case
const cochee = await page.$eval('[data-testid=accept-cgv]', (el) => el.checked);
expect.ok(cochee, 'La case des conditions générales devrait être cochée');
```

14.8.4 Clavier

```javascript
await page.keyboard.press('Enter');      // Valider
await page.keyboard.press('Tab');        // Champ suivant
await page.keyboard.press('Escape');     // Fermer une fenêtre modale
await page.keyboard.press('ArrowDown');  // Parcourir une liste de suggestions
```

Ces événements sont générés par le navigateur, comme s'ils provenaient d'un vrai clavier. Il n'est plus nécessaire de recourir à des bibliothèques de la page, comme jQuery, pour simuler une touche, ce qu'imposait l'ancienne génération.

14.8.5 Survol

```javascript
// Déplier un menu qui s'ouvre au survol
await page.locator('[data-testid=menu-compte]').hover();
await page.locator('[data-testid=menu-commandes]').click();
```

14.8.6 Éléments facultatifs : bandeaux et fenêtres d'information

Un bandeau de consentement aux cookies ou une fenêtre promotionnelle peut apparaître lors de la première visite, ce qui est toujours le cas avec le profil vierge de la sonde. S'il est présent, il faut le fermer. Son absence ne doit pas faire échouer le scénario :

```javascript
await step('Fermer le bandeau de consentement', async () => {
  // Attente courte : le bandeau est facultatif
  const bandeau = await page
    .waitForSelector('[data-testid=cookie-refuse]', { visible: true, timeout: 3000 })
    .catch(() => null);
  if (bandeau) await bandeau.click();
});
```

> ✓ **Conseil** — Lorsque c'est possible, préférez **refuser** les cookies non essentiels plutôt que les accepter. La sonde ne fausse ainsi pas les statistiques d'audience, et le parcours se rapproche de celui d'un utilisateur prudent.

14.9 Authentification

14.9.1 Formulaire de connexion

Le cas le plus courant est un formulaire avec identifiant et mot de passe. Les valeurs proviennent toujours de l'objet `secrets` :

```javascript
await step('Se connecter', async () => {
  await page.goto(`${params.baseUrl}/connexion`, { waitUntil: 'domcontentloaded' });
  await page.locator('[data-testid=login-email]').fill(secrets.BOUTIQUE_USER);
  await page.locator('[data-testid=login-password]').fill(secrets.BOUTIQUE_PASSWORD);
  await Promise.all([
    page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
    page.locator('[data-testid=login-submit]').click(),
  ]);
  // Indispensable : prouver que la connexion a réussi
  await expect.visible(page, '[data-testid=account-menu]');
});
```

La vérification finale est essentielle. De nombreuses applications répondent avec un code HTTP 200 **même lorsque l'authentification échoue**, en réaffichant le formulaire accompagné d'un message d'erreur. Sans vérification de l'espace authentifié, le scénario conclurait à tort au succès.

Les secrets utilisés sont créés par un administrateur et **déclarés** dans le contrôle (liste `secrets`). Leur création et leur rotation sont décrites au chapitre 9.

14.9.2 Authentification HTTP de base

Certaines applications internes demandent une authentification par le navigateur (fenêtre de connexion du navigateur, mécanisme HTTP *Basic*). Elle se configure avant la navigation :

```javascript
await page.authenticate({ username: secrets.INTRANET_USER, password: secrets.INTRANET_PASSWORD });
await page.goto(`${params.baseUrl}/`, { waitUntil: 'domcontentloaded' });
```

14.9.3 Fournisseurs d'identité et authentification unique

Avec une authentification unique (SSO, *Single Sign-On*), l'utilisateur est redirigé vers un fournisseur d'identité, puis renvoyé vers l'application. Le scénario suit le même chemin :

1. ouverture de l'application ;
2. attente du formulaire du fournisseur d'identité, sur un autre domaine ;
3. saisie des identifiants du compte de test ;
4. attente du retour vers l'application et vérification de l'espace authentifié.

Le navigateur de la sonde doit pouvoir joindre le fournisseur d'identité (chapitre 7). Nommez distinctement l'étape de connexion au fournisseur : si celui-ci est indisponible, le résultat le montrera clairement.

14.9.4 Second facteur d'authentification

Un scénario ne peut pas recevoir un code SMS, valider une notification sur un téléphone ou utiliser une clé physique. Pour superviser une application protégée par un second facteur, convenez avec les équipes sécurité d'un **compte de test dédié**, exempté du second facteur. Encadrez-le par des mesures compensatoires : droits minimaux, restriction aux adresses IP de la sonde, mot de passe long renouvelé régulièrement, surveillance des connexions.

> **Avertissement** — Ne cherchez jamais à contourner un second facteur ou un CAPTCHA, par exemple avec un service de résolution automatique. Outre le risque juridique et contractuel, cela affaiblirait la sécurité de l'application que vous supervisez.

14.9.5 Sessions

Le profil vierge impose une authentification à **chaque exécution** (chapitre 13, § 13.3.3). C'est voulu : la connexion fait partie du service à vérifier, et une session conservée masquerait une panne du mécanisme d'authentification. Si plusieurs scénarios d'un même plan d'exécution nécessitent une session, chacun se connecte. Le chapitre 16, § 16.7 détaille la gestion des sessions et des comptes de test.

14.10 Gérer l'attente et les délais

La gestion des attentes est la principale source de fiabilité ou de fragilité d'un scénario. Une application web n'est jamais instantanée : chaque action déclenche des chargements, des appels réseau et des mises à jour de l'affichage. Le scénario doit attendre **le bon événement**, ni trop tôt ni trop longtemps.

14.10.1 Principe : attendre un événement, jamais une durée

À faire	À proscrire
Attendre qu'un élément apparaisse	Attendre 5 secondes « par sécurité »
Attendre la réponse d'une API	Supposer que la réponse est arrivée

À faire	À proscrire
Attendre qu'une condition soit vraie dans la page	Recommencer l'action en boucle

Une pause fixe est **toujours** mauvaise. Trop courte, elle provoque des échecs aléatoires lorsque l'application est un peu lente. Trop longue, elle allonge chaque exécution, fausse les mesures de durée et occupe inutilement le moteur de scénarios.

> **Information** — L'ancienne fonction de pause de Puppeteer (`page.waitForTimeout`) a été supprimée des versions récentes. N'écrivez pas de pause avec `setTimeout` non plus : utilisez l'une des attentes ci-dessous.

14.10.2 Les attentes disponibles

Besoin	Instruction
Agir sur un élément dès qu'il est prêt	`page.locator(sélecteur).click()` , `.fill()` , `.hover()` : attente intégrée
Attendre qu'un élément soit présent sans agir	`page.locator(sélecteur).wait()`
Attendre qu'un élément soit visible	`page.waitForSelector(sélecteur, { visible: true })`
Attendre qu'un élément disparaisse (indicateur de chargement)	`page.waitForSelector(sélecteur, { hidden: true })`
Attendre une nouvelle page	`page.waitForNavigation({ waitUntil: 'domcontentloaded' })` , armée avant le clic
Attendre la réponse d'une requête	`page.waitForResponse(condition)` , armée avant l'action
Attendre une condition calculée dans la page	`page.waitForFunction(fonction, options, …arguments)`

14.10.3 Attendre une réponse réseau

Lorsqu'une action déclenche un appel d'API, attendre la réponse permet de vérifier son code et de mesurer précisément le temps du service.

```javascript
const [reponse] = await Promise.all([
  page.waitForResponse((r) => r.url().includes('/api/produits') && r.request().method() === 'GET'),
  page.locator('[data-testid=search-submit]').click(),
]);
expect.equal(reponse.status(), 200, "Code HTTP de l'API de recherche");
```

La condition passée à `waitForResponse` doit être **précise**, par exemple le chemin et la méthode. Une condition trop large risque de retenir une autre requête de la page.

14.10.4 Attendre une condition

`waitForFunction` évalue une fonction dans la page jusqu'à ce qu'elle renvoie une valeur vraie. Les valeurs du scénario sont transmises en arguments après les options :

```javascript
// Attendre que le compteur du panier ait augmenté de 1
await page.waitForFunction(
  (avant) => Number(document.querySelector('[data-testid=cart-count]').textContent) === avant + 1,
  { timeout: 10000 },
  articlesAvant,
);
```

14.10.5 Hiérarchie des délais

Trois niveaux de délais s'appliquent à un scénario :

Niveau	Défini par	Valeur par défaut	Effet du dépassement
Timeout du contrôle	`timeout` du contrôle	Selon le contrôle (exemple : `60s`)	Le moteur interrompt l'exécution ; état `CRITICAL`, `error.type = timeout`
Délai de navigation	Moteur de scénarios ; option `timeout` de `goto` et `waitForNavigation`	30 s	Exception, `error.type = navigation_error`
Délai d'attente d'un élément ou d'une condition	Moteur ; option `timeout` des attentes ; `locator().setTimeout(ms)`	10 s	Exception, `error.type = element_not_found` ou `assertion_failed`

Règles pratiques :

- fixez le timeout du contrôle à **deux à trois fois la durée normale** du parcours (§ 8.4), et au moins 30 secondes ;
- n'augmentez le délai d'une attente que pour une opération connue pour être lente (génération d'un document, recherche complexe), et seulement pour cette attente ;
- ne cherchez pas à masquer une lenteur par un délai long : utilisez plutôt les seuils de durée (`thresholds.warningDuration`), qui signalent la lenteur par l'état `WARNING` sans faire échouer le scénario (chapitre 8).

```javascript
// Délai spécifique à une opération lente connue
await page.locator('[data-testid=export-ready]').setTimeout(30000).wait();
```

14.11 Assertions

Une **assertion** est une vérification explicite. Si elle n'est pas satisfaite, elle lève une exception et l'étape échoue avec un message lisible. Un scénario sans assertion ne vérifie presque rien : il prouve seulement que les éléments cliqués existaient.

14.11.1 Fonctions fournies

Fonction	Échoue si…	Exemple
`expect.visible(cible, sélecteur, { timeout })`	L'élément n'est pas visible dans le délai	`await expect.visible(page, '[data-testid=account-menu]')`

Fonction	Échoue si…	Exemple
`expect.hidden(cible, sélecteur, { timeout })`	L'élément reste visible au-delà du délai	`await expect.hidden(page, '[data-testid=loading]')`
`expect.text(cible, sélecteur, attendu, { timeout, exact })`	Le texte ne contient pas `attendu` (ou ne lui est pas égal si `exact: true`) dans le délai	`await expect.text(page, '[data-testid=welcome]', 'Bienvenue')`
`expect.url(cible, motif)`	L'adresse courante ne contient pas le motif (chaîne) ou ne lui correspond pas (expression régulière)	`expect.url(page, /\/compte$/)`
`expect.ok(condition, message)`	La condition est fausse	`expect.ok(lignes.length > 0, 'Panier vide')`
`expect.equal(réel, attendu, message)`	Les deux valeurs ne sont pas strictement égales	`expect.equal(total, 179, 'Total du panier')`

Les fonctions `visible`, `hidden` et `text` sont asynchrones, car elles attendent : elles doivent être précédées de `await`. `url`, `ok` et `equal` vérifient immédiatement. Le paramètre `cible` est généralement `page`, mais peut aussi être un cadre (`Frame`) obtenu à partir d'une iframe (§ 14.13.1).

14.11.2 Écrire de bonnes assertions

- **Vérifiez le résultat, pas l'action.** Après un ajout au panier, vérifiez le contenu du panier, pas seulement l'affichage du message « Produit ajouté ».
- **Rédigez des messages utiles.** Le message apparaît dans les résultats et dans les alertes. « Total du panier : attendu 179, obtenu 89,5 » permet un diagnostic immédiat, contrairement à « Erreur ».
- **Comparez des valeurs techniques.** Lisez un attribut (`data-value="89.50"`) plutôt qu'un texte formaté (« 89,50 € »), qui dépend de la langue et de la mise en forme.
- **Tolérez ce qui peut légitimement varier.** Un nombre de résultats de recherche peut évoluer avec le catalogue : vérifiez un minimum plutôt qu'une valeur exacte.

```javascript
// Assertion avec un message de diagnostic explicite
const totalAffiche = Number(await page.$eval('[data-testid=cart-total]', (el) => el.dataset.value));
const totalAttendu = Math.round(prixUnitaire * quantite * 100) / 100;
expect.equal(totalAffiche, totalAttendu, 'Total du panier');
```

14.12 Traitement des erreurs

14.12.1 Le principe : laisser l'exception remonter

Dans un scénario, une erreur est une **exception**. Toute exception non interceptée dans un `step()` :

1. marque l'étape en échec ;
2. arrête le scénario ;
3. déclenche la collecte des éléments de diagnostic (capture d'écran, console, requêtes en erreur) ;
4. détermine l'état du contrôle et le type d'erreur.

Ce comportement convient dans la très grande majorité des cas. **N'interceptez pas les exceptions pour les ignorer** : un `try/catch` qui avale une erreur transforme une panne en succès apparent.

```javascript
// À NE PAS FAIRE : l'échec est masqué, le scénario conclut à tort au succès
try {
  await expect.visible(page, '[data-testid=account-menu]');
} catch (e) {
  // ignoré
}
```

14.12.2 Cas légitimes d'interception

Interceptez une erreur seulement lorsque l'échec est **normal** et prévu par le parcours, par exemple un élément facultatif (§ 14.8.6) :

```javascript
const bandeau = await page
  .waitForSelector('[data-testid=promo-close]', { visible: true, timeout: 3000 })
  .catch(() => null); // l'absence du bandeau n'est pas une erreur
if (bandeau) await bandeau.click();
```

Pour **enrichir** un message d'erreur, interceptez l'exception, puis levez-en une nouvelle :

```javascript
try {
  await page.locator('[data-testid=pay-test-card]').click();
} catch (e) {
  throw new Error(`Module de paiement de test indisponible : ${e.message}`);
}
```

14.12.3 Types d'erreur et états

Le moteur de scénarios classe chaque échec dans la liste fermée des types d'erreur de la sonde, dont la référence complète figure au chapitre 17 (§ 17.5.2). Pour les scénarios, les plus fréquents sont :

Type (`error.type`)	Situation typique	État
`assertion_failed`	Une assertion `expect` n'est pas satisfaite	`CRITICAL`
`element_not_found`	Un élément attendu par une action n'est jamais apparu	`CRITICAL`
`navigation_error`	La page n'a pas pu être chargée (erreur réseau, délai de navigation)	`CRITICAL`
`dns_error`, `connection_refused`, `tls_error`	Erreur réseau identifiée lors d'une navigation	`CRITICAL`
`timeout`	Le timeout global du contrôle est dépassé	`CRITICAL`
`script_error`	Erreur de programmation dans le script (variable non définie, fonction inexistante)	`CRITICAL`
`browser_crash`	Le navigateur s'est arrêté de façon inattendue	`UNKNOWN`
`secret_missing`	Un secret déclaré dans `secrets` est absent de la sonde : le scénario ne démarre pas	`UNKNOWN`

L'état `UNKNOWN` signale un problème de la sonde ou de l'outil d'exécution, et non de l'application supervisée. Il ne doit pas être confondu avec une panne applicative (chapitre 17).

14.12.4 Journaliser utilement

`log.info()` et `log.warn()` ajoutent des informations au journal de l'exécution, consultable avec le résultat. Utilisez-les pour les valeurs qui faciliteront le diagnostic : nombre de résultats, numéro de commande de test, valeur lue.

```javascript
log.info(`${resultats.length} résultat(s) pour « ${terme} »`);
```

Ne journalisez jamais une donnée personnelle ni une valeur sensible. Les valeurs des secrets déclarés sont masquées automatiquement, mais une donnée affichée par l'application (adresse, numéro de client) ne l'est pas.

14.13 Iframes, fenêtres, boîtes de dialogue et téléchargements

14.13.1 Iframes

Une **iframe** est une page intégrée dans une autre : formulaire de paiement, module de contact, carte, contenu d'un service tiers. Ses éléments appartiennent à un document distinct. Un sélecteur exécuté sur la page principale ne les trouve pas.

En mode scripté, on obtient d'abord le cadre (`Frame`), puis on travaille dans ce cadre :

```javascript
// Récupérer le cadre de l'iframe identifiée par #widget-contact
const cadre = await (await page.waitForSelector('#widget-contact')).contentFrame();
await expect.visible(cadre, '[data-testid=contact-form]');
await cadre.locator('[data-testid=contact-message]').fill('Message de test');
```

En mode déclaratif, la propriété `frame` de l'étape indique le sélecteur de l'iframe :

```json
{ "name": "Vérifier le formulaire de contact", "action": "assertVisible", "frame": "#widget-contact",
    "selector": "[data-testid=contact-form]" }
```

> **Information** — Les iframes de paiement des prestataires spécialisés sont souvent conçues pour résister à l'automatisation, pour des raisons de sécurité. En environnement de test, utilisez le mode de paiement simulé fourni par le prestataire ou par votre application (chapitre 15, exemple 7).

14.13.2 Nouvelles fenêtres et onglets

Un lien ouvert dans un nouvel onglet (`target="_blank"`) ou une fenêtre ouverte par un script crée une nouvelle page. On l'intercepte avec l'événement `popup`, armé avant le clic :

```javascript
const [fenetre] = await Promise.all([
  new Promise((resoudre) => page.once('popup', resoudre)),
  page.locator('[data-testid=open-terms]').click(),
]);
await fenetre.waitForSelector('[data-testid=terms-content]', { visible: true });
await fenetre.close(); // la fenêtre secondaire peut être fermée par le script
```

La page principale (`page`) reste ouverte et utilisable. Le moteur de scénarios ferme toutes les fenêtres en fin d'exécution.

14.13.3 Boîtes de dialogue

Les boîtes de dialogue JavaScript (`alert`, `confirm`, `prompt`) bloquent la page tant qu'elles ne sont pas fermées. Préparez leur traitement avant l'action qui les déclenche :

```javascript
// Accepter la prochaine boîte de confirmation
page.once('dialog', (dialogue) => dialogue.accept());
await page.locator('[data-testid=cart-clear]').click();
```

Pour refuser, utilisez `dialogue.dismiss()`. `dialogue.message()` renvoie le texte affiché, qui peut être vérifié.

14.13.4 Téléchargements

Le téléchargement de fichiers par le navigateur n'est pas un usage prévu des scénarios de la sonde : le moteur de scénarios **refuse les téléchargements**, afin que les fichiers ne s'accumulent pas sur la carte microSD. Pour vérifier qu'un document est disponible (facture de test, export), vérifiez plutôt la **réponse** du serveur :

```javascript
const [reponse] = await Promise.all([
  page.waitForResponse((r) => r.url().includes('/factures/') && r.request().method() === 'GET'),
  page.locator('[data-testid=invoice-download]').click(),
]);
expect.equal(reponse.status(), 200, 'Téléchargement de la facture de test');
expect.ok((reponse.headers()['content-type'] || '').includes('application/pdf'), 'Type de document
    inattendu');
```

14.14 Fermeture et libération des ressources

Avec la sonde, le **moteur de scénarios** est responsable du cycle de vie du navigateur :

- il lance un navigateur neuf avant l'appel du script ;
- il le ferme après le script, que l'exécution ait réussi, échoué ou dépassé son timeout ;
- en cas de blocage, il arrête de force le navigateur et tous ses processus ;
- il détruit le profil temporaire (cookies, cache, stockage local).

Le script ne doit donc **ni lancer ni fermer** le navigateur. N'appelez pas `browser.close()` ni `page.close()` sur la page principale. Les fenêtres secondaires que vous ouvrez peuvent être fermées par le script (§ 14.13.2), mais ce n'est pas obligatoire.

Cette répartition simplifie l'écriture : contrairement à un script Puppeteer autonome, un scénario n'a pas besoin de blocs `try/finally` pour garantir la fermeture du navigateur. Elle garantit aussi qu'aucun processus Chromium orphelin ne subsiste sur la sonde, ce qui est essentiel sur un appareil disposant d'une mémoire limitée.

> ! **Attention** — Un scénario Puppeteer trouvé sur Internet ou produit par un outil d'enregistrement commence généralement par `puppeteer.launch()` et se termine par `browser.close()`. Ces lignes doivent être **supprimées**, et le reste du code placé dans la fonction exportée, organisé en étapes.

14.15 Tester sur poste de travail avec le lanceur local

14.15.1 Présentation

Le dossier `exemples/puppeteer/` contient un **lanceur local** (`lanceur-local.mjs`). Cet outil pédagogique reproduit sur un poste de travail le contrat d'exécution de la sonde : objet de contexte, étapes, assertions, timeout global, capture en cas d'échec, résultat JSON. Il permet de mettre au point un scénario rapidement, en observant le navigateur, avant de le déployer.

> **Information** — Le lanceur local est **distinct du produit**. Il utilise Puppeteer standard et le navigateur téléchargé par Puppeteer sur le poste. Il n'offre ni l'isolation, ni le chiffrement des secrets, ni la planification, ni la télémétrie de la sonde, et sa classification des erreurs est une approximation. La validation finale se fait toujours par une exécution d'essai sur la sonde (§ 14.16).

14.15.2 Procédure

Objectif : exécuter un scénario sur un poste de travail et lire son résultat.

Prérequis : Node.js 22.12 ou version ultérieure (minimum exigé par Puppeteer 25) ; accès à l'application à tester (ou à l'application de démonstration fournie).

Étapes

1. Installez les dépendances des exemples :

```bash
cd exemples/puppeteer
npm install
```

2. Si vous utilisez l'application de démonstration, démarrez-la dans un premier terminal :

```bash
npm run demo
```

3. Dans un second terminal, définissez les secrets dans des variables d'environnement, jamais sur la ligne de commande :

```bash
# Identifiant du compte de test fictif de l'application de démonstration
export BOUTIQUE_USER='test-supervision@example.com'
# Saisie masquée du mot de passe (Demo-2026! pour l'application de démonstration) :
# il n'apparaît ni à l'écran ni dans l'historique du shell
read -rs BOUTIQUE_PASSWORD && export BOUTIQUE_PASSWORD
```

4. Exécutez le scénario en indiquant ses paramètres et le **nom** des secrets à transmettre :

```bash
node lanceur-local.mjs scenarios/03-authentification.mjs \
  --param baseUrl=http://localhost:8080 \
  --secret BOUTIQUE_USER --secret BOUTIQUE_PASSWORD
```

5. Pour observer le navigateur pendant la mise au point, ajoutez `--visible`.

Résultat attendu : le lanceur affiche le résultat au format JSON et se termine avec le code 0 si l'état est `OK`.

Option	Rôle
`--param nom=valeur`	Paramètre non sensible (répétable)

Option	Rôle
`--secret NOM`	Secret lu dans la variable d'environnement `NOM` (répétable)
`--timeout 60s`	Délai global du scénario (60 s par défaut), hors lancement et fermeture du navigateur (voir ci-dessous)
`--avertissement 8s`	Durée au-delà de laquelle un succès devient `WARNING`
`--bloquer domaine`	Bloque les requêtes vers un domaine (répétable)
`--visible`	Affiche le navigateur
`--sortie dossier`	Dossier des artefacts (par défaut `resultats/`)

Le lanceur accepte aussi un **contrôle déclaratif** complet au format JSON. Les paramètres et la liste des secrets sont alors lus dans le fichier :

```bash
node lanceur-local.mjs scenarios/03-authentification.json
```

Phases et délais du lanceur. Une exécution comprend trois phases, bornées séparément :

Phase	Contenu	Borne
1. Lancement	Démarrage du navigateur, puis ouverture et préparation de la page (taille d'affichage, délais par défaut, écouteurs de diagnostic)	Démarrage : délai de lancement de Puppeteer, 30 s. Préparation : délai propre de 30 s ; en cas d'erreur ou de dépassement, le navigateur est fermé et le lanceur s'arrête avec le code 3 (`UNKNOWN`)
2. Scénario	Exécution de la fonction du scénario, étape par étape	Délai global (`--timeout`, ou `timeout` du contrôle déclaratif)
3. Fermeture	Capture d'écran en cas d'échec, puis fermeture du navigateur	5 s pour la capture, 5 s pour la fermeture, puis arrêt forcé du processus

Le délai global ne couvre donc que la phase 2, qui est aussi la seule mesurée par `dureeMs` : la durée totale de la commande peut le dépasser de la durée du lancement et de la fermeture. Les minuteries des délais sont annulées dès qu'elles deviennent inutiles.

> **Information** — Le délai global est mis en œuvre par une mise en concurrence (`Promise.race`) entre le scénario et une minuterie. Cette construction **n'annule pas** les opérations déjà entamées par le scénario : lorsque le délai expire, le lanceur cesse d'attendre, refuse toute nouvelle étape, puis ferme le navigateur, et c'est cette fermeture qui interrompt les opérations encore en cours. Le moteur de scénarios de la sonde applique ses propres délais (chapitre 8).

14.15.3 Lire le résultat

```json
{
  "id": "01M7TPKDNETWDPE3DP1EJTTDK8",
  "checkId": "03-authentification",
  "trigger": "dryRun",
  "status": "OK",
  "startedAt": "2026-11-19T20:05:08.783Z",
  "durationMs": 344,
  "error": null,
  "steps": [
    { "index": 0, "name": "Ouvrir la page de connexion", "status": "OK", "startedOffsetMs": 1,
    "durationMs": 151 },
    { "index": 1, "name": "Saisir les identifiants", "status": "OK", "startedOffsetMs": 152,
    "durationMs": 89 },
    { "index": 2, "name": "Valider le formulaire", "status": "OK", "startedOffsetMs": 241,
    "durationMs": 94 },
    { "index": 3, "name": "Vérifier l'accès à l'espace client", "status": "OK", "startedOffsetMs":
    335, "durationMs": 9 }
  ],
  "artifacts": [],
  "log": []
}
```

Ce résultat est au **format du lanceur local**. Il reprend la structure des résultats de la sonde (chapitre 17), mais n'est pas conforme au schéma `run-result` du produit. Ses écarts sont les suivants :

Écart	Format du lanceur local	Résultat de la sonde
`log`	Présent : journal des avertissements collectés pendant l'exécution	Absent : le schéma n'admet pas de champ supplémentaire ; les journaux sont des artefacts (`kind: console`)
`checkRevision`	Absent : le lanceur ne gère pas de révisions de configuration	Obligatoire
`checkId`	Nom du fichier, ou `id` du contrôle déclaratif	Identifiant du contrôle sur la sonde
`attempt` , `planRunId` , `measurements` , `trace`	Absents	Présents selon le contexte d'exécution
`trigger`	Toujours `dryRun`	Selon le déclenchement
`error.type`	Classification approchée	Classification du moteur de scénarios

Un outil qui valide les résultats selon le schéma de la sonde (intégration, tests d'un consommateur de l'API) doit donc utiliser l'exemple complet du chapitre 17, § 17.4, et non la sortie du lanceur local. En cas d'échec, `error` indique le type, l'étape et le message. Une capture d'écran `failure.png` et le journal `console.json` sont enregistrés dans `resultats/<identifiant>/`. Le code de sortie suit la convention Nagios : 0 `OK` , 1 `WARNING` , 2 `CRITICAL` , 3 `UNKNOWN` .

La durée mesurée commence après le lancement du navigateur et s'arrête à la fin du scénario, avant la fermeture. Elle reflète donc le temps de l'application, et non celui du démarrage ou de la fermeture de Chromium.

> ✓ **Conseil** — Testez systématiquement le **cas d'échec** de votre scénario : modifiez temporairement une valeur attendue, ou utilisez un mot de passe erroné dans votre variable d'environnement, et vérifiez que le scénario échoue à la bonne étape avec un message compréhensible. Un scénario qui ne sait pas échouer ne protège de rien.

14.16 Déployer un scénario sur la sonde

14.16.1 Scénario déclaratif

Objectif : mettre en service un scénario déclaratif.

Prérequis : rôle `operator` ou `admin` ; édition Enterprise ; secrets nécessaires créés par un administrateur (chapitre 9) ; flux réseau ouverts vers l'application et ses ressources (chapitre 7).

Étapes

1. Créez le contrôle désactivé, par l'interface d'administration ou par l'API :

   ```bash
   # Fonction sapi définie au chapitre 9, § 9.3.3 (certificat vérifié, jeton lu dans un fichier)
   sapi -X POST -H 'Content-Type: application/json' \
     --data @demo-connexion.json "$SONDE/api/v1/checks"
   ```

   Le fichier `demo-connexion.json` reprend l'exemple `scenarios/03-authentification.json`. Avant l'envoi, adaptez `params.baseUrl` : dans l'exemple, `http://localhost:8080` désigne le poste de test, alors que sur la sonde, `localhost` désigne la sonde elle-même.

   Pour créer le contrôle désactivé, positionnez `"enabled": false` dans le fichier. La sonde valide la structure et refuse un contrôle invalide avec un message d'erreur détaillé (chapitre 9).

2. Lancez une exécution d'essai :

   ```bash
   sapi -X POST -D - "$SONDE/api/v1/checks/demo-connexion/runs?dryRun=true"
   ```

   L'exécution d'essai réalise le scénario sans modifier l'état du contrôle et sans transmettre de télémétrie. La réponse indique l'adresse de suivi de l'exécution.

3. Consultez le résultat de l'essai (étapes, durée, erreur éventuelle, capture d'écran) dans l'interface ou par `GET /api/v1/runs/{runId}` (chapitre 17).
4. Si le résultat est conforme, activez le contrôle : `POST /api/v1/checks/demo-connexion/enable`.

Vérification : après la première échéance planifiée, le contrôle apparaît avec l'état `OK` dans le tableau de bord.

14.16.2 Scénario scripté

Objectif : mettre en service un scénario scripté.

Prérequis : rôle `admin` ; édition Enterprise ; script testé sur poste de travail (§ 14.15) ; secrets créés.

Étapes

1. **Téléversez le script** par l'interface d'administration ou par `POST /api/v1/scripts`. Chaque requête envoie un seul module `.mjs`, avec le type de contenu `text/javascript`, dans la limite de 1 Mo. La sonde renvoie une **référence immuable** de la forme `nom@sha256:empreinte`, par exemple `boutique-connexion@sha256:4f1c9e2ab7d3` (empreinte abrégée ici ; la référence réelle comporte 64 caractères hexadécimaux).
2. **Créez le contrôle** de type `browser`, en mode `script`, désactivé, en indiquant cette référence dans

`browser.scriptRef` :

```json
{
  "id": "boutique-connexion",
  "name": "Boutique : connexion du compte de test",
  "type": "browser",
  "enabled": false,
  "schedule": { "interval": "5m", "jitter": "30s" },
  "timeout": "60s",
  "secrets": ["BOUTIQUE_USER", "BOUTIQUE_PASSWORD"],
  "browser": {
    "mode": "script",
    "scriptRef": "boutique-connexion@sha256:4f1c9e2ab7d3",
    "params": { "baseUrl": "https://boutique.example.com" },
    "captureOnFailure": true
  }
}
```

3. **Lancez une exécution d'essai** (`?dryRun=true`), comme au § 14.16.1.
4. **Analysez le résultat.** Un écart entre le poste de travail et la sonde provient le plus souvent :
   - d'une ressource que la sonde ne peut pas joindre (pare-feu, proxy, résolution DNS) ;
   - d'une version différente de Puppeteer ou de Chromium (§ 13.8.2) ;
   - de performances plus modestes, qui nécessitent un délai adapté.
5. **Activez le contrôle**, puis ajoutez-le si nécessaire à un plan d'exécution (chapitre 12).

Vérification : le contrôle s'exécute selon sa planification ; les résultats et la télémétrie apparaissent (chapitres 17 et 18).

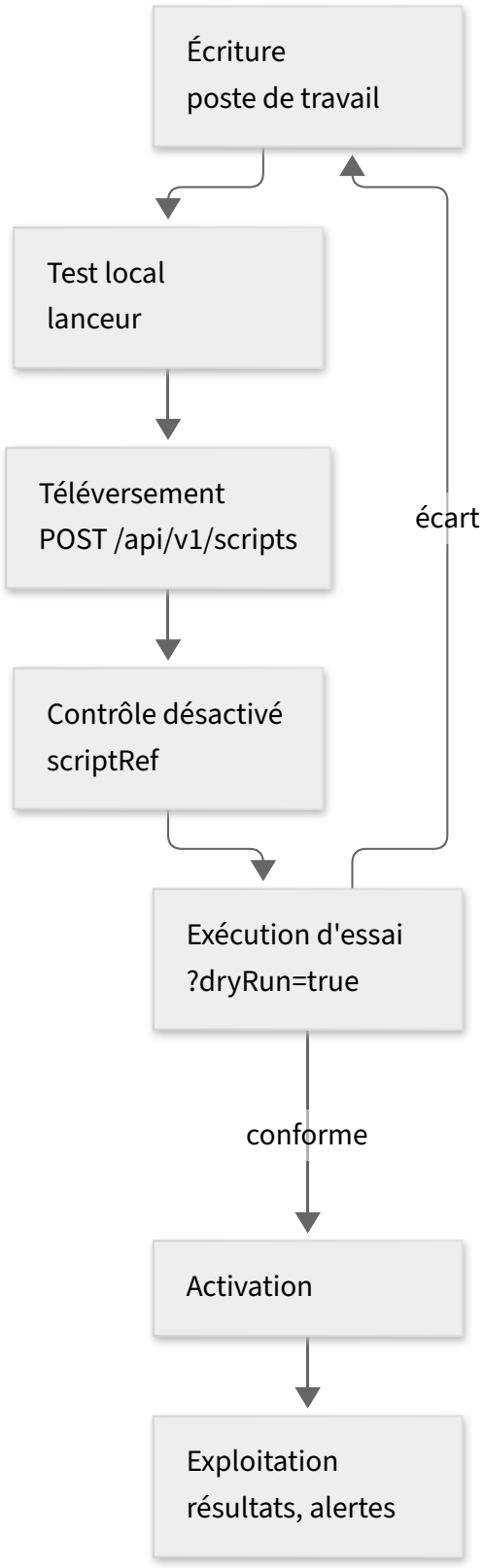


Schéma 14-2 — Cycle de mise en service d'un scénario scripté

14.17 Cycle de vie et versionnage des scripts

14.17.1 Références immuables

Chaque téléversement d'un script crée une **nouvelle version**, identifiée par l'empreinte de son contenu. Un contrôle désigne toujours une version précise (`scriptRef`). Ce mécanisme apporte trois garanties :

- **aucun changement implicite** : téléverser une nouvelle version ne modifie aucun contrôle existant ;
- **traçabilité** : chaque résultat est rattaché à la révision du contrôle, et donc à la version exacte du script exécuté ;
- **retour arrière immédiat** : il suffit de remettre l'ancienne référence dans le contrôle.

14.17.2 Procédure de mise à jour d'un scénario scripté

1. Modifiez le script dans votre gestionnaire de versions (Git recommandé).
2. Testez-le sur poste de travail, y compris le cas d'échec (§ 14.15).
3. Téléversez la nouvelle version et notez la référence renvoyée.
4. Pour un changement important, créez un **contrôle temporaire** désactivé utilisant la nouvelle référence, et lancez une exécution d'essai.
5. Mettez à jour `browser.scriptRef` dans le contrôle de production (`PATCH /api/v1/checks/{id}`). Cette modification requiert le rôle `admin` (§ 14.1).
6. Surveillez les exécutions suivantes. En cas de régression, restaurez la référence précédente.

> ✓ **Conseil** — Inscrivez la référence `nom@sha256:empreinte` renvoyée par la sonde dans le message de version de votre gestionnaire de code. Le lien entre le code source et ce qui s'exécute sur la sonde reste ainsi vérifiable à tout moment.

14.17.3 Coordination avec les mises en production

Une évolution de l'application supervisée peut rendre un scénario obsolète : nouveau libellé, nouvelle étape dans le tunnel, sélecteur supprimé. Le chapitre 16, § 16.15 décrit l'organisation recommandée pour maintenir les scénarios au rythme des mises en production, et la manière de suspendre temporairement un scénario pendant une intervention planifiée.

14.17.4 Retrait d'un scénario

Lorsqu'un parcours n'existe plus :

1. désactivez le contrôle et retirez-le des plans d'exécution ;
2. supprimez le contrôle lorsque son historique n'est plus utile (l'historique déjà enregistré est conservé selon la rétention, chapitre 8, § 8.12) ;
3. supprimez les secrets qui ne sont plus utilisés par aucun contrôle ;
4. informez les équipes qui reçoivent les alertes ou les tableaux de bord associés.

Chapitre 15 — Exemples progressifs

Ce chapitre présente huit scénarios complets, du plus simple au plus élaboré. Chacun répond à un besoin de supervision courant et met en œuvre une notion nouvelle. Pour chaque exemple sont présentés l'objectif, les prérequis, le code commenté, les explications, la version déclarative lorsqu'elle est possible, le résultat attendu, les variantes utiles et les erreurs courantes.

Tous les exemples ont été exécutés et vérifiés sur un poste de travail, avec le lanceur local décrit au chapitre 14, § 14.15, et *Chrome for Testing*. La sonde utilise, elle, le Chromium des dépôts Raspberry Pi OS : avant de déployer un exemple, effectuez une exécution d'essai sur votre sonde (chapitre 14, § 14.16).

15.1 Présentation des exemples

15.1.1 L'application de démonstration

Les exemples ciblent la **Boutique Démo**, une application web **fictive** fournie dans le dossier `exemples/puppeteer/application-demo/`. Elle s'exécute sur votre poste de travail (`http://localhost:8080`) et reproduit les fonctions d'un site marchand simple :

Fonction	Adresse	Particularité utile aux exemples
Accueil	`/`	Produits à la une ; bandeau « environnement de test »
Catalogue	`/catalogue`	Liste des produits
Fiche produit	`/produit/P100` … `/produit/P400`	Ajout au panier par appel d'API, avec un délai simulé
Recherche	`/recherche`	Résultats affichés par JavaScript après un appel d'API
Connexion	`/connexion`	Répond HTTP 200 même en cas d'échec d'authentification
Espace client	`/compte`	Accessible après authentification
Aide	`/aide`	Formulaire de contact dans une iframe
Panier	`/panier`	Lignes, quantités, total
Commande de test	`/commande`	Tunnel en trois étapes, paiement **simulé**, annulation possible

Le compte de test est **fictif** : `test-supervision@example.com`, mot de passe `Demo-2026!`. Tous les éléments utiles portent un attribut `data-testid`, comme le recommande le chapitre 16.

Pour les besoins de l'exemple 8, l'application peut simuler deux **pannes fonctionnelles**. Elles s'activent par un paramètre d'adresse réservé à la démonstration :

Paramètre	Panne simulée
`?panne=panier`	L'ajout au panier affiche « Produit ajouté » mais n'enregistre rien
`?panne=prix`	Le total du panier ignore la quantité

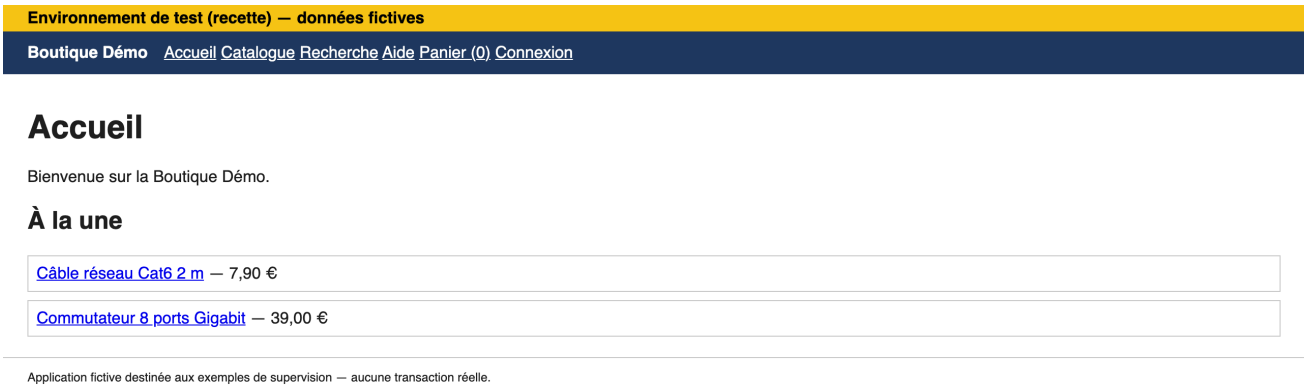


Figure 15-1 — Page d'accueil de la Boutique Démo avec le bandeau « Environnement de test »

15.1.2 Vue d'ensemble

Exemple	Objectif	Notion introduite	Déclaratif	Fichiers
1	Page accessible	`goto`, code HTTP, titre	Oui	`01-page-accessible.*`
2	Élément présent	Attente et vérification d'un contenu	Oui	`02-element-present.*`
3	Authentification	Secrets, formulaire, navigation par clic	Oui	`03-authentification.*`
3 bis	Refus d'authentification	Vérifier un échec attendu	Oui	`03b-authentification-refusee.*`
4	Plusieurs pages	Enchaînement d'étapes, iframe	Oui	`04-navigation-pages.*`
5	Moteur de recherche	Attente d'une réponse d'API, pertinence	Partiel	`05-recherche.*`
6	Ajout au panier	Attente d'une condition, contrôle d'état	Oui	`06-ajout-panier.*`
7	Commande de test	Garde-fous, parcours long, nettoyage	Non	`07-parcours-commande.mjs`
8	Erreur fonctionnelle	Calcul du résultat attendu, indices techniques	Non	`08-detection-erreur-fonctionnelle.mjs`

15.1.3 Exécuter les exemples

Les fichiers des exemples sont livrés dans l'archive `synovex-exemples.zip`, fournie avec les PDF de la documentation et téléchargeable depuis la page d'accueil de la documentation en ligne ; décompressée, elle crée le dossier `exemples/`. La procédure complète est décrite au chapitre 14, § 14.15, et dans le fichier `exemples/puppeteer/README.md`. En résumé :

```bash
cd exemples/puppeteer
npm install
npm run demo                       # premier terminal : application fictive
export BOUTIQUE_USER='test-supervision@example.com'
read -rs BOUTIQUE_PASSWORD && export BOUTIQUE_PASSWORD   # saisie masquée : Demo-2026! (compte
    fictif)
node lanceur-local.mjs scenarios/01-page-accessible.mjs --param baseUrl=http://localhost:8080
```

> **Information** — Par défaut, l'application de démonstration n'écoute que sur la **boucle locale** (`127.0.0.1`) : elle n'est joignable que depuis le poste qui l'exécute, même si celui-ci est raccordé à un réseau partagé (salle de formation, réseau d'entreprise). C'est le mode à utiliser pour tous les exemples de ce chapitre.

Mise à disposition volontaire sur un réseau de recette. Pour qu'une sonde de test exécute les exemples contre la Boutique Démo, celle-ci doit être joignable depuis le réseau. Cette exposition est une opération distincte et volontaire :

1. Obtenez l'accord du responsable du réseau : utilisez uniquement un réseau de recette ou de formation autorisé, jamais un réseau de production ni un réseau ouvert (Wi-Fi public, invités).
2. Relevez l'adresse IP du poste sur ce réseau.
3. Démarrez l'application en indiquant cette adresse dans la variable d'environnement `HOTE` (exemple fictif : `HOTE=192.0.2.80 npm run demo`). Le serveur affiche alors un avertissement rappelant que l'application est exposée.
4. Limitez si possible les accès au port de l'application (8080 par défaut) à l'adresse de la sonde de test, par le pare-feu du poste.
5. Dans les scénarios, remplacez `http://localhost:8080` par `http://<adresse-du-poste>:8080` (paramètre `baseUrl`).
6. Arrêtez l'application (Ctrl+C) dès la fin des essais.

> **Avertissement** — La Boutique Démo n'a aucune sécurité : son compte de test et son mot de passe sont publiés dans cette documentation, ses échanges ne sont pas chiffrés et elle ne journalise pas les accès. Ne l'exposez jamais sur toutes les interfaces du poste (`HOTE=0.0.0.0`) ni sur un réseau que vous ne maîtrisez pas.

Pour déployer un exemple sur une sonde, remplacez `baseUrl` par l'adresse de votre application, adaptez les sélecteurs, puis suivez le chapitre 14, § 14.16.

> **Information** — Les blocs « Résultat attendu » de ce chapitre sont des **extraits (projection)** de la sortie du lanceur local : ils ne reprennent que les champs utiles à l'explication (`checkId`, `status`, `durationMs`, `error`, `steps`, `artifacts`). La sortie complète du lanceur et ses écarts avec le format de la sonde sont décrits au chapitre 14, § 14.15.3 ; l'exemple de résultat intégral de la sonde figure au chapitre 17, § 17.4.

> **Avertissement** — Ces scénarios sont écrits pour l'application de démonstration. Ne les exécutez jamais tels quels contre une application réelle, et jamais contre une application de production sans l'accord de ses responsables. Les exemples 6 et 7 modifient des données (panier, commande).

15.2 Exemple 1 — Vérifier qu'une page web est accessible

15.2.1 Objectif

Vérifier que la page d'accueil se charge dans un navigateur, répond avec un code HTTP de succès et provient bien de l'application attendue.

C'est le scénario le plus simple. Il va déjà au-delà d'un contrôle `http` : la page est traitée par un vrai navigateur, et la vérification du titre écarte les fausses réussites, comme une page de maintenance ou la page d'erreur d'un proxy servies avec un code 200.

15.2.2 Prérequis

- Application de démonstration démarrée.
- Paramètre `baseUrl`.

15.2.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/01-page-accessible.mjs` :

```javascript
// Exemple 1 — Vérifier qu'une page web est accessible.
// Application cible : « Boutique Démo » (fictive). Paramètre : baseUrl.
// Contrat : le moteur fournit page, step, params et expect ; il lance et ferme le navigateur.

export default async function scenario({ page, step, params, expect }) {
  await step("Ouvrir la page d'accueil", async () => {
    // page.goto() renvoie la réponse HTTP du document principal.
    const reponse = await page.goto(`${params.baseUrl}/`, { waitUntil: 'domcontentloaded' });
    // Une page accessible doit répondre avec un code 2xx.
    expect.ok(reponse && reponse.ok(), `Code HTTP inattendu : ${reponse ? reponse.status() : 'aucune
    réponse'}`);
  });

  await step('Vérifier le titre de la page', async () => {
    // Le titre confirme que la bonne application a répondu (et non une page d'erreur d'un proxy).
    const titre = await page.title();
    expect.ok(titre.includes('Boutique Démo'), `Titre inattendu : « ${titre} »`);
  });
}
```

15.2.4 Explications

- `page.goto()` renvoie la **réponse HTTP du document principal**. `reponse.ok()` est vrai pour un code compris entre 200 et 299. La réponse peut être nulle dans de rares cas (navigation vers la même adresse avec une ancre) : le test `reponse && …` couvre ce cas.
- `waitUntil: 'domcontentloaded'` considère la page chargée dès que le document est analysé, sans attendre toutes les images. C'est la valeur recommandée (chapitre 14, § 14.6.1).
- Les deux étapes apparaissent séparément dans le résultat. Une page lente se distingue ainsi d'une page erronée.

15.2.5 Version déclarative

Fichier `exemples/puppeteer/scenarios/01-page-accessible.json` :

```json
{
  "id": "demo-accueil-accessible",
  "name": "Boutique Démo : page d'accueil accessible",
  "type": "browser",
  "schedule": { "interval": "5m", "jitter": "30s" },
  "timeout": "30s",
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080" },
    "steps": [
      { "name": "Ouvrir la page d'accueil", "action": "goto", "url": "${param:baseUrl}/" },
      { "name": "Vérifier le titre de la page", "action": "assertTitle", "operator": "contains",
    "value": "Boutique Démo" }
    ]
  }
}
```

En mode déclaratif, l'action `goto` échoue si la page ne peut pas être chargée (erreur réseau, délai dépassé). La vérification explicite du code HTTP n'est pas disponible dans ce mode : l'assertion sur le titre la remplace utilement.

15.2.6 Résultat attendu

```json
{
  "checkId": "01-page-accessible",
  "status": "OK",
  "durationMs": 76,
  "error": null,
  "steps": [
    { "index": 0, "name": "Ouvrir la page d'accueil", "status": "OK", "durationMs": 71 },
    { "index": 1, "name": "Vérifier le titre de la page", "status": "OK", "durationMs": 5 }
  ]
}
```

Les durées dépendent de l'environnement. Elles sont ici très courtes, car l'application fonctionne sur le poste de travail lui-même.

15.2.7 Variantes

- **Seuil de performance** : ajoutez `"thresholds": { "warningDuration": "3s" }` au contrôle pour obtenir un état `WARNING` lorsque la page devient lente.
- **Page interne** : la même structure vérifie toute page importante (page de connexion, page de statut publique).

15.2.8 Erreurs courantes

Symptôme	Cause probable	Correction
`navigation_error` sur l'étape 1	Adresse erronée ; application arrêtée ; flux bloqué par le pare-feu	Vérifier `baseUrl` et les flux (chapitre 7)
`connection_refused`	Aucun service n'écoute sur l'adresse et le port indiqués	Vérifier le port et l'état de l'application
`tls_error`	Certificat expiré, ou émis par une autorité interne non reconnue par la sonde	Renouveler le certificat ; importer l'autorité interne (chapitre 7)

Symptôme	Cause probable	Correction
Titre inattendu	Page de maintenance, page d'un proxy, redirection vers un portail captif	Examiner la capture d'écran (chapitre 17)

15.3 Exemple 2 — Vérifier la présence d'un élément attendu

15.3.1 Objectif

Vérifier que la page affiche son **contenu utile** : le message d'accueil et au moins un produit à la une. Une page peut se charger correctement alors que la zone principale reste vide, par exemple lorsque le service qui fournit les produits est en panne.

15.3.2 Prérequis

Identiques à l'exemple 1.

15.3.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/02-element-present.mjs` :

JAVASCRIPT

```
// Exemple 2 — Vérifier la présence d'un élément attendu.
// On ne se contente pas d'une réponse HTTP : on vérifie que le contenu utile est affiché.

export default async function scenario({ page, step, params, expect }) {
  await step("Ouvrir la page d'accueil", async () => {
    await page.goto(`${params.baseUrl}/`, { waitUntil: 'domcontentloaded' });
  });

  await step("Vérifier le message d'accueil", async () => {
    // Attente explicite de l'élément, puis contrôle de son texte.
    await expect.text(page, '[data-testid=welcome]', 'Bienvenue');
  });

  await step('Vérifier la présence des produits à la une', async () => {
    // expect.visible() attend qu'au moins un produit soit visible :
    // il garantit à lui seul la présence d'un produit à la une.
    await expect.visible(page, '[data-testid=featured-product]');
  });
}
```

15.3.4 Explications

- `expect.text()` **attend** que l'élément existe et que son texte contienne la valeur attendue, dans la limite du délai par défaut. Si le contenu est affiché par un script après le chargement, l'assertion patiente sans pause fixe.
- `expect.visible()` attend qu'au moins un élément correspondant soit **visible**. Un élément présent dans le document mais masqué ne suffit pas.
- Pour exiger **plusieurs** éléments, comptez-les après cette attente : `page.$$()` renvoie la liste des éléments correspondants, et `expect.ok(produits.length >= 3, …)` vérifie la quantité minimale. Un seuil de 1 serait redondant : `expect.visible()` échouerait avant.

15.3.5 Version déclarative

Fichier `exemples/puppeteer/scenarios/02-element-present.json` :

```json
{
  "id": "demo-accueil-contenu",
  "name": "Boutique Démo : contenu de la page d'accueil",
  "type": "browser",
  "schedule": { "interval": "5m" },
  "timeout": "30s",
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080" },
    "steps": [
      { "name": "Ouvrir la page d'accueil", "action": "goto", "url": "${param:baseUrl}/" },
      { "name": "Vérifier le message d'accueil", "action": "assertText", "selector": "[data-
    testid=welcome]", "operator": "contains", "value": "Bienvenue" },
      { "name": "Vérifier la présence des produits à la une", "action": "assertVisible", "selector":
    "[data-testid=featured-product]" }
    ]
  }
}
```

15.3.6 Résultat attendu

État `OK`, trois étapes `OK`. Si la zone des produits est vide, l'étape « Vérifier la présence des produits à la une » échoue avec un message de la forme :

```json
{
  "status": "CRITICAL",
  "error": {
    "type": "assertion_failed",
    "step": "Vérifier la présence des produits à la une",
    "message": "Élément [data-testid=featured-product] non visible après 10000 ms"
  }
}
```

15.3.7 Variantes

- **Absence d'un message d'erreur** : `await expect.hidden(page, '.message-erreur-global')` vérifie qu'aucun bandeau d'incident n'est affiché.
- **Contenu dynamique** : pour un élément dont le texte varie (date, prix), vérifiez sa présence et sa forme plutôt que sa valeur exacte, par exemple avec une expression régulière en mode déclaratif (`"operator": "matches"`).

15.3.8 Erreurs courantes

Symptôme	Cause probable	Correction
Échec après une mise en production, alors que la page semble correcte	Le sélecteur ne correspond plus (attribut renommé)	Mettre à jour le sélecteur ; convenir d'attributs `data-testid` stables (chapitre 16)
Échec aléatoire	Contenu affiché tardivement, au-delà du délai	Vérifier la lenteur réelle ; ajuster le délai de cette seule assertion
Le scénario réussit alors que la page est vide	Sélecteur trop large, qui trouve un autre élément	Rendre le sélecteur spécifique ; vérifier un texte

15.4 Exemple 3 — S'authentifier sur une application

15.4.1 Objectif

Vérifier que le compte de test peut se connecter et accéder à son espace client. C'est l'un des scénarios les plus utiles : une authentification défaillante rend l'application inutilisable pour tous les utilisateurs, alors que les pages publiques restent accessibles et que les contrôles techniques restent au vert.

15.4.2 Prérequis

- Paramètre `baseUrl`.
- Secrets `BOUTIQUE_USER` et `BOUTIQUE_PASSWORD`, créés sur la sonde par un administrateur et déclarés dans le contrôle (chapitre 9). Sur poste de travail, ce sont des variables d'environnement.
- Un **compte de test dédié**, aux droits minimaux, convenu avec les responsables de l'application (chapitre 16, § 16.7).

15.4.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/03-authentification.mjs` :

```javascript
// Exemple 3 — S'authentifier sur une application avec un compte de test.
// Secrets déclarés par le contrôle : BOUTIQUE_USER, BOUTIQUE_PASSWORD.
// Les valeurs ne figurent jamais dans le script : elles sont fournies par le moteur.

export default async function scenario({ page, step, secrets, params, expect }) {
  await step('Ouvrir la page de connexion', async () => {
    await page.goto(`${params.baseUrl}/connexion`, { waitUntil: 'domcontentloaded' });
    await expect.visible(page, '[data-testid=login-form]');
  });

  await step('Saisir les identifiants', async () => {
    // locator().fill() attend que le champ soit disponible, le vide, puis saisit la valeur.
    await page.locator('[data-testid=login-email]').fill(secrets.BOUTIQUE_USER);
    await page.locator('[data-testid=login-password]').fill(secrets.BOUTIQUE_PASSWORD);
  });

  await step('Valider le formulaire', async () => {
    // Le clic déclenche une navigation : on l'attend en même temps que le clic,
    // pour ne pas manquer un chargement très rapide.
    await Promise.all([
      page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
      page.locator('[data-testid=login-submit]').click(),
    ]);
  });

  await step("Vérifier l'accès à l'espace client", async () => {
    // L'application répond HTTP 200 même en cas d'échec d'authentification :
    // seule la présence de l'espace client prouve la réussite.
    await expect.visible(page, '[data-testid=account-menu]');
    expect.url(page, '/compte');
    await expect.text(page, '[data-testid=account-name]', 'Bonjour');
  });
}
```

15.4.4 Explications

- **Aucun identifiant dans le code.** `secrets.BOUTIQUE_USER` et `secrets.BOUTIQUE_PASSWORD` sont fournis par le moteur de scénarios, uniquement parce qu'ils sont déclarés dans le contrôle. Leurs valeurs sont masquées dans

les journaux.

- **Navigation armée avant le clic.** `Promise.all([page.waitForNavigation(…), …click()])` garantit que le chargement de la page suivante ne sera pas manqué (chapitre 14, § 14.6.2).
- **Vérification de la réussite.** L'application de démonstration, comme beaucoup d'applications réelles, répond HTTP 200 même lorsque l'authentification échoue. Seule la présence de l'espace client (`account-menu`), associée à l'adresse `/compte`, prouve la connexion.

15.4.5 Version déclarative

Fichier `exemples/puppeteer/scenarios/03-authentification.json` :

```json
{
  "id": "demo-connexion",
  "name": "Boutique Démo : connexion du compte de test",
  "type": "browser",
  "schedule": { "interval": "5m", "jitter": "30s" },
  "timeout": "45s",
  "retry": { "attempts": 1, "interval": "30s" },
  "secrets": ["BOUTIQUE_USER", "BOUTIQUE_PASSWORD"],
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080" },
    "captureOnFailure": true,
    "steps": [
      { "name": "Ouvrir la page de connexion", "action": "goto", "url": "${param:baseUrl}/connexion"
    },
      { "name": "Saisir l'identifiant", "action": "fill", "selector": "[data-testid=login-email]",
    "value": "${secret:BOUTIQUE_USER}" },
      { "name": "Saisir le mot de passe", "action": "fill", "selector": "[data-testid=login-
    password]", "value": "${secret:BOUTIQUE_PASSWORD}" },
      { "name": "Valider le formulaire", "action": "click", "selector": "[data-testid=login-submit]",
    "waitForNavigation": true },
      { "name": "Vérifier l'accès à l'espace client", "action": "assertVisible", "selector": "[data-
    testid=account-menu]" },
      { "name": "Vérifier l'URL de l'espace client", "action": "assertUrl", "operator": "contains",
    "value": "/compte" }
    ]
  }
}
```

La propriété `retry` du contrôle autorise une nouvelle tentative après 30 secondes avant de conclure à l'état `CRITICAL`. Un incident ponctuel, comme une perte de paquets réseau, ne déclenche ainsi pas d'alerte (chapitre 8).

15.4.6 Résultat attendu

Résultat obtenu avec le lanceur local :

```json
{
  "checkId": "03-authentification",
  "status": "OK",
  "durationMs": 344,
  "error": null,
  "steps": [
    { "index": 0, "name": "Ouvrir la page de connexion", "status": "OK", "startedOffsetMs": 1,
    "durationMs": 151 },
    { "index": 1, "name": "Saisir les identifiants", "status": "OK", "startedOffsetMs": 152,
    "durationMs": 89 },
    { "index": 2, "name": "Valider le formulaire", "status": "OK", "startedOffsetMs": 241,
    "durationMs": 94 },
    { "index": 3, "name": "Vérifier l'accès à l'espace client", "status": "OK", "startedOffsetMs":
    335, "durationMs": 9 }
  ]
}
```

Avec un mot de passe erroné, le scénario échoue à la dernière étape :

```json
{
  "status": "CRITICAL",
  "error": {
    "type": "assertion_failed",
    "step": "Vérifier l'accès à l'espace client",
    "message": "Élément [data-testid=account-menu] non visible après 10000 ms"
  }
}
```

15.4.7 Variantes

- **Authentification unique (SSO)** : ajoutez une étape « Se connecter au fournisseur d'identité » entre l'ouverture de l'application et la vérification de l'espace client (chapitre 14, § 14.9.3).
- **Déconnexion** : ajoutez une étape finale qui clique sur « Se déconnecter » et vérifie le retour à la page publique. Le mécanisme de déconnexion est ainsi vérifié, et la session est libérée côté serveur.

15.4.8 Erreurs courantes

Symptôme	Cause probable	Correction
`UNKNOWN` avant toute étape	Secret non déclaré dans `secrets` (mode scripté)	Créer le secret et l'ajouter à la liste `secrets` (chapitre 9)
Échec à « Vérifier l'accès à l'espace client »	Mot de passe expiré ou modifié ; compte verrouillé	Vérifier le compte avec les responsables ; mettre à jour le secret
Échecs répétés puis verrouillage du compte	Les nouvelles tentatives multiplient les échecs d'authentification	Limiter `retry.attempts` ; convenir d'une politique de verrouillage adaptée au compte de test
Échec sur une autre étape après une mise en production	Formulaire modifié	Mettre à jour les sélecteurs

15.4.9 Exemple complémentaire 3 bis — Vérifier le refus d'une authentification invalide

Ce scénario reprend l'intention d'un scénario historique de la génération précédente : soumettre volontairement des identifiants **invalides** et vérifier que l'application les refuse en affichant son message d'erreur. Il complète l'exemple 3. Il vérifie que le mécanisme d'authentification n'accepte pas n'importe quel identifiant, et que la page d'erreur fonctionne.

Fichier `exemples/puppeteer/scenarios/03b-authentification-refusee.mjs` :

JAVASCRIPT

```
// Exemple 3 bis — Vérifier qu'une authentification invalide est correctement refusée.
// Reprend l'intention d'un scénario historique : on soumet volontairement des
// identifiants erronés et l'on vérifie que l'application affiche son message d'erreur.
// Le compte utilisé est fictif et n'existe pas : aucun secret n'est nécessaire.
// Attention : sur une application réelle, des échecs répétés peuvent verrouiller un
// compte ou alerter l'équipe sécurité ; utilisez un identifiant inexistant convenu avec elle.

export default async function scenario({ page, step, params, expect }) {
  await step('Ouvrir la page de connexion', async () => {
    await page.goto(`${params.baseUrl}/connexion`, { waitUntil: 'domcontentloaded' });
  });

  await step('Soumettre des identifiants invalides', async () => {
    await page.locator('[data-testid=login-email]').fill('compte-inexistant@example.com');
    await page.locator('[data-testid=login-password]').fill('mot-de-passe-volontairement-faux');
    await Promise.all([
      page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
      page.locator('[data-testid=login-submit]').click(),
    ]);
  });

  await step("Vérifier le message d'erreur", async () => {
    // Le résultat attendu est un refus : le message doit apparaître…
    await expect.text(page, '[data-testid=login-error]', "Erreur d'authentification");
    // … et l'espace client ne doit pas être accessible.
    await expect.hidden(page, '[data-testid=account-menu]');
  });
}
```

Version déclarative :

Fichier `exemples/puppeteer/scenarios/03b-authentification-refusee.json` :

```json
{
  "id": "demo-connexion-refusee",
  "name": "Boutique Démo : refus d'une authentification invalide",
  "type": "browser",
  "schedule": { "interval": "15m" },
  "timeout": "30s",
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080" },
    "steps": [
      { "name": "Ouvrir la page de connexion", "action": "goto", "url": "${param:baseUrl}/connexion"
    },
      { "name": "Saisir un identifiant inexistant", "action": "fill", "selector": "[data-
    testid=login-email]", "value": "compte-inexistant@example.com" },
      { "name": "Saisir un mot de passe erroné", "action": "fill", "selector": "[data-testid=login-
    password]", "value": "mot-de-passe-volontairement-faux" },
      { "name": "Valider le formulaire", "action": "click", "selector": "[data-testid=login-submit]",
    "waitForNavigation": true },
      { "name": "Vérifier le message d'erreur", "action": "assertText", "selector": "[data-
    testid=login-error]", "operator": "contains", "value": "Erreur d'authentification" },
      { "name": "Vérifier l'absence d'accès", "action": "assertHidden", "selector": "[data-
    testid=account-menu]" }
    ]
  }
}
```

Résultat attendu : état `OK`, car le refus est le comportement attendu. Si l'application acceptait l'identifiant inexistant, ou n'affichait plus son message d'erreur, le scénario passerait à l'état `CRITICAL`.

> **Attention** — Sur une application réelle, des échecs d'authentification répétés peuvent déclencher des alertes de sécurité ou verrouiller un compte existant. Convenez avec l'équipe sécurité d'un identifiant **inexistant** réservé à ce scénario, et d'une fréquence modérée (toutes les 15 minutes ou plus).

15.5 Exemple 4 — Parcourir plusieurs pages

15.5.1 Objectif

Vérifier que les pages principales s'enchaînent : accueil, catalogue, fiche produit, aide. Le formulaire de contact, intégré dans une **iframe**, doit aussi être disponible. Chaque page constitue une étape : le résultat indique laquelle est lente ou en échec.

15.5.2 Prérequis

- Paramètre `baseUrl`, et paramètre facultatif `produit` (par défaut `P200`).

15.5.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/04-navigation-pages.mjs` :

```javascript
// Exemple 4 — Parcourir plusieurs pages, y compris un contenu placé dans une iframe.
// Chaque page visitée constitue une étape : le résultat indique laquelle est lente ou en échec.

export default async function scenario({ page, step, params, expect }) {
  // Clic sur un lien suivi de l'attente de la nouvelle page.
```

```
  const cliquerEtAttendre = (selecteur) =>
    Promise.all([
      page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
      page.locator(selecteur).click(),
    ]);

  await step("Ouvrir la page d'accueil", async () => {
    await page.goto(`${params.baseUrl}/`, { waitUntil: 'domcontentloaded' });
    await expect.visible(page, '[data-testid=welcome]');
  });

  await step('Ouvrir le catalogue', async () => {
    await cliquerEtAttendre('[data-testid=nav-catalogue]');
    await expect.text(page, '[data-testid=page-title]', 'Catalogue', { exact: true });
    const articles = await page.$$('[data-testid=catalog-item]');
    expect.ok(articles.length > 0, 'Le catalogue est vide');
  });

  await step('Ouvrir une fiche produit', async () => {
    // Le produit visé est un paramètre : le scénario reste valable si le catalogue évolue.
    await cliquerEtAttendre(`[data-testid=product-link-${params.produit || 'P200'}]`);
    await expect.visible(page, '[data-testid=product-name]');
    await expect.visible(page, '[data-testid=add-to-cart]');
  });

  await step("Ouvrir la page d'aide", async () => {
    await cliquerEtAttendre('[data-testid=nav-aide]');
    await expect.visible(page, '[data-testid=contact-frame]');
  });

  await step('Vérifier le formulaire de contact (iframe)', async () => {
    // Les éléments d'une iframe appartiennent à un autre document :
    // on obtient d'abord l'objet Frame, puis on travaille dans ce cadre.
    const cadre = await (await page.waitForSelector('#widget-contact')).contentFrame();
    await expect.visible(cadre, '[data-testid=contact-form]');
    await expect.visible(cadre, '[data-testid=contact-submit]');
    // Aucun message n'est envoyé : la supervision ne doit pas créer de faux contacts.
  });
}
```

15.5.4 Explications

- **Fonction utilitaire.** `cliquerEtAttendre()` évite de répéter le motif « attente de navigation et clic » à chaque page. Elle est déclarée hors des étapes, car elle n'agit pas sur la page.
- **Navigation par les liens**, et non par des adresses directes. Le scénario vérifie ainsi aussi que les liens de navigation fonctionnent, comme le ferait un utilisateur.
- **Iframe.** Le formulaire de contact appartient à un autre document. `contentFrame()` donne accès à ce document, sur lequel les assertions s'appliquent normalement (chapitre 14, § 14.13.1).
- **Aucun envoi.** Le scénario vérifie la présence du formulaire sans l'envoyer. La supervision ne doit pas créer de faux messages dans les outils de support.

15.5.5 Version déclarative

Fichier `exemples/puppeteer/scenarios/04-navigation-pages.json` :

```json
{
  "id": "demo-navigation",
  "name": "Boutique Démo : navigation entre les pages principales",
  "type": "browser",
  "schedule": { "interval": "10m" },
  "timeout": "60s",
  "thresholds": { "warningDuration": "15s" },
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080" },
    "steps": [
      { "name": "Ouvrir la page d'accueil", "action": "goto", "url": "${param:baseUrl}/" },
      { "name": "Ouvrir le catalogue", "action": "click", "selector": "[data-testid=nav-catalogue]",
    "waitForNavigation": true },
      { "name": "Vérifier le catalogue", "action": "assertVisible", "selector": "[data-
    testid=catalog-item]" },
      { "name": "Ouvrir une fiche produit", "action": "click", "selector": "[data-testid=product-
    link-P200]", "waitForNavigation": true },
      { "name": "Vérifier la fiche produit", "action": "assertVisible", "selector": "[data-
    testid=add-to-cart]" },
      { "name": "Ouvrir la page d'aide", "action": "click", "selector": "[data-testid=nav-aide]",
    "waitForNavigation": true },
      { "name": "Vérifier le formulaire de contact (iframe)", "action": "assertVisible", "frame":
    "#widget-contact", "selector": "[data-testid=contact-form]" }
    ]
  }
}
```

La propriété `frame` de la dernière étape désigne l'iframe dans laquelle l'élément doit être recherché.

15.5.6 Résultat attendu

État `OK`, cinq étapes (sept en version déclarative). Le seuil `warningDuration` de 15 secondes, défini dans la version déclarative, signale par un état `WARNING` un parcours anormalement lent.

15.5.7 Variantes

- **Pages critiques métier** : remplacez le parcours par les pages qui comptent pour vos utilisateurs (conditions générales, page de contact, foire aux questions).
- **Vérification du menu** : ajoutez des assertions sur la présence de chaque entrée du menu principal.

15.5.8 Erreurs courantes

Symptôme	Cause probable	Correction
`navigation_error` sur un clic	Le lien n'entraîne pas de chargement de page (application monopage)	Remplacer `waitForNavigation` par l'attente d'un élément de la nouvelle vue (chapitre 14, § 14.6.3)
`element_not_found` dans l'iframe	Sélecteur recherché dans la page principale au lieu du cadre ; iframe chargée lentement	Utiliser le cadre (`contentFrame`) ; attendre l'élément dans le cadre
Iframe d'un domaine tiers inaccessible	Flux réseau non autorisé depuis la sonde	Autoriser le domaine de l'iframe (chapitre 7)

15.6 Exemple 5 — Vérifier le fonctionnement d'un moteur de recherche

15.6.1 Objectif

Vérifier que la recherche renvoie des résultats **pertinents** : un nombre minimal de résultats, qui contiennent tous le terme recherché. La recherche de l'application est exécutée par un appel d'API depuis la page : le scénario attend la réponse de cette API et contrôle son code.

15.6.2 Prérequis

- Paramètres `baseUrl`, `terme` (par défaut « câble ») et `minimum` (par défaut 1).
- Un terme dont les résultats sont **stables** dans le temps, convenu avec les responsables du catalogue.

15.6.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/05-recherche.mjs` :

JAVASCRIPT

```
// Exemple 5 — Vérifier le fonctionnement d'un moteur de recherche.
// La recherche de l'application est exécutée en JavaScript par un appel d'API :
// on attend la réponse de cette API, puis l'affichage des résultats.
// Paramètres : baseUrl, terme (défaut « câble »), minimum (défaut 1).

export default async function scenario({ page, step, params, expect, log }) {
  const terme = params.terme || 'câble';
  const minimum = Number(params.minimum || 1);

  await step('Ouvrir la page de recherche', async () => {
    await page.goto(`${params.baseUrl}/recherche`, { waitUntil: 'domcontentloaded' });
    await expect.visible(page, '[data-testid=search-input]');
  });

  await step('Lancer la recherche', async () => {
    await page.locator('[data-testid=search-input]').fill(terme);
    // On arme l'attente de la réponse AVANT le clic, puis on clique.
    const [reponse] = await Promise.all([
      page.waitForResponse((r) => r.url().includes('/api/produits') && r.request().method() ===
    'GET'),
      page.locator('[data-testid=search-submit]').click(),
    ]);
    expect.equal(reponse.status(), 200, "Code HTTP de l'API de recherche");
  });

  await step('Vérifier les résultats', async () => {
    // Le compteur n'apparaît qu'une fois les résultats affichés.
    await expect.visible(page, '[data-testid=search-count]');
    const resultats = await page.$$eval('[data-testid=search-result]', (els) => els.map((e) =>
    e.textContent.trim()));
    log.info(`${resultats.length} résultat(s) pour « ${terme} »`);
    expect.ok(resultats.length >= minimum, `${resultats.length} résultat(s) pour « ${terme} »,
    ${minimum} attendu(s) au minimum`);
    // Pertinence : chaque résultat doit contenir le terme recherché.
    const horsSujet = resultats.filter((t) => !t.toLowerCase().includes(terme.toLowerCase()));
    expect.ok(horsSujet.length === 0, `Résultats sans rapport avec « ${terme} » : ${horsSujet.join(',
    ')}`);
  });
}
```

15.6.4 Explications

- **Attente d'une réponse d'API.** `page.waitForResponse()` est armée **avant** le clic, dans un `Promise.all`. La condition est précise : chemin `/api/produits` et méthode `GET`. Le code de la réponse est vérifié explicitement. Si l'API répond en erreur, le message l'indique, ce qui oriente immédiatement le diagnostic vers le service de recherche plutôt que vers l'interface.
- **Lecture groupée.** `page.$$eval()` lit en une seule opération le texte de tous les résultats.
- **Pertinence.** Vérifier qu'il existe des résultats ne suffit pas : un moteur défaillant peut renvoyer tout le catalogue. Le scénario vérifie que chaque résultat contient le terme.
- **Journalisation.** Le nombre de résultats est journalisé. En cas d'anomalie, cette information est disponible avec le résultat.

15.6.5 Version déclarative

Fichier `exemples/puppeteer/scenarios/05-recherche.json` :

```json
{
  "id": "demo-recherche",
  "name": "Boutique Démo : moteur de recherche",
  "type": "browser",
  "schedule": { "interval": "10m" },
  "timeout": "45s",
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080", "terme": "câble" },
    "steps": [
      { "name": "Ouvrir la page de recherche", "action": "goto", "url": "${param:baseUrl}/recherche"
    },
      { "name": "Saisir le terme recherché", "action": "fill", "selector": "[data-testid=search-
    input]", "value": "${param:terme}" },
      { "name": "Lancer la recherche", "action": "click", "selector": "[data-testid=search-submit]"
    },
      { "name": "Vérifier l'affichage des résultats", "action": "assertVisible", "selector": "[data-
    testid=search-result]" },
      { "name": "Vérifier le compteur de résultats", "action": "assertText", "selector": "[data-
    testid=search-count]", "operator": "matches", "value": "^[1-9][0-9]* résultat" }
    ]
  }
}
```

La version déclarative vérifie l'affichage des résultats et la forme du compteur. Elle ne peut ni contrôler la réponse de l'API, ni vérifier la pertinence de chaque résultat : c'est un exemple typique de besoin qui justifie le mode scripté.

15.6.6 Résultat attendu

État `OK`. Le journal de l'exécution contient :

```json
"log": [ { "niveau": "INFO", "message": "2 résultat(s) pour « câble »" } ]
```

Avec un terme sans résultat (`--param terme=introuvable`), le scénario échoue à l'étape attendue :

```json
{
  "status": "CRITICAL",
  "error": {
    "type": "assertion_failed",
    "step": "Vérifier les résultats",
    "message": "0 résultat(s) pour « introuvable », 1 attendu(s) au minimum"
  }
}
```

15.6.7 Variantes

- **Temps de réponse du moteur** : ajoutez un seuil `warningDuration` au contrôle, ou mesurez la durée entre le clic et la réponse et journalisez-la.
- **Validation par la touche Entrée** : remplacez le clic par `page.keyboard.press('Enter')` dans le `Promise.all`, pour vérifier le comportement le plus courant chez les utilisateurs.
- **Suggestions** : vérifiez la liste de suggestions qui s'affiche pendant la frappe, en saisissant avec `page.keyboard.type()` (chapitre 14, § 14.8.2).

15.6.8 Erreurs courantes

Symptôme	Cause probable	Correction
`timeout` ou `element_not_found` à « Lancer la recherche »	La condition de `waitForResponse` ne correspond à aucune requête (chemin d'API différent)	Identifier la requête dans l'onglet *Network* des outils de développement
Faux échec après une évolution du catalogue	Le terme choisi n'a plus de résultats	Convenir d'un terme et d'un produit de référence pérennes
Résultats « hors sujet »	Recherche floue ou synonymes volontaires	Adapter le critère de pertinence

15.7 Exemple 6 — Ajouter un produit au panier

15.7.1 Objectif

Vérifier que l'ajout d'un produit au panier fonctionne **réellement** : l'API répond, le compteur augmente et le produit figure dans le panier. Cet exemple reprend la situation qui a motivé la supervision applicative (chapitre 2) : un site qui s'affiche correctement, mais sur lequel il est impossible d'ajouter un produit au panier.

15.7.2 Prérequis

- Paramètres `baseUrl` et `produit` (par défaut `P100`).
- Un **produit de test** convenu avec le métier : toujours disponible, sans incidence sur les stocks réels ni sur les statistiques de vente.

15.7.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/06-ajout-panier.mjs` :

```javascript
// Exemple 6 — Ajouter un produit au panier.
// Le navigateur démarre avec un profil vierge : le panier est donc vide au départ.
// Paramètres : baseUrl, produit (défaut P100).
```

```
export default async function scenario({ page, step, params, expect }) {
  const produit = params.produit || 'P100';
  let articlesAvant = 0;

  await step('Ouvrir la fiche produit', async () => {
    await page.goto(`${params.baseUrl}/produit/${produit}`, { waitUntil: 'domcontentloaded' });
    await expect.visible(page, '[data-testid=add-to-cart]');
    articlesAvant = Number(await page.$eval('[data-testid=cart-count]', (el) => el.textContent));
  });

  await step('Ajouter le produit au panier', async () => {
    // Le bouton déclenche un appel POST à l'API du panier ; on attend sa réponse.
    const [reponse] = await Promise.all([
      page.waitForResponse((r) => r.url().includes('/api/panier') && r.request().method() ===
    'POST'),
      page.locator('[data-testid=add-to-cart]').click(),
    ]);
    expect.equal(reponse.status(), 200, "Code HTTP de l'API du panier");
    await expect.visible(page, '[data-testid=add-to-cart-message]');
  });

  await step('Vérifier le compteur du panier', async () => {
    // Attente d'une condition calculée dans la page plutôt que d'un délai fixe.
    await page.waitForFunction(
      (avant) => Number(document.querySelector('[data-testid=cart-count]').textContent) === avant +
    1,
      {},
      articlesAvant,
    );
  });

  await step('Vérifier le contenu du panier', async () => {
    await page.goto(`${params.baseUrl}/panier`, { waitUntil: 'domcontentloaded' });
    await expect.visible(page, `[data-testid=cart-line][data-product-id=${produit}]`);
  });
}
```

15.7.4 Explications

- **Lecture de l'état initial.** Le nombre d'articles est lu avant l'ajout. Le profil du navigateur est vierge, et le panier est donc vide en principe, mais le scénario ne fait pas cette hypothèse.
- **Attente d'une condition.** `page.waitForFunction()` attend que le compteur ait augmenté de un. La valeur `articlesAvant` est transmise en argument, car le code de la fonction s'exécute dans la page (chapitre 14, § 14.10.4).
- **Vérification du résultat, pas du message.** Le message « Produit ajouté » est vérifié, puis le panier est ouvert pour s'assurer que la ligne du produit existe. Seule cette dernière vérification prouve que l'ajout a été enregistré.

15.7.5 Version déclarative

Fichier `exemples/puppeteer/scenarios/06-ajout-panier.json` :

```json
{
  "id": "demo-ajout-panier",
  "name": "Boutique Démo : ajout d'un produit au panier",
  "type": "browser",
  "schedule": { "interval": "10m" },
  "timeout": "45s",
  "browser": {
    "mode": "steps",
    "params": { "baseUrl": "http://localhost:8080" },
    "steps": [
      { "name": "Ouvrir la fiche produit", "action": "goto", "url": "${param:baseUrl}/produit/P100"
    },
      { "name": "Ajouter le produit au panier", "action": "click", "selector": "[data-testid=add-to-
    cart]" },
      { "name": "Vérifier la confirmation d'ajout", "action": "assertText", "selector": "[data-
    testid=add-to-cart-message]", "operator": "contains", "value": "ajouté" },
      { "name": "Vérifier le compteur du panier", "action": "assertText", "selector": "[data-
    testid=cart-count]", "operator": "eq", "value": "1" },
      { "name": "Ouvrir le panier", "action": "goto", "url": "${param:baseUrl}/panier" },
      { "name": "Vérifier le contenu du panier", "action": "assertVisible", "selector": "[data-
    testid=cart-line][data-product-id=P100]" }
    ]
  }
}
```

15.7.6 Résultat attendu

État `OK`, quatre étapes (six en version déclarative).

15.7.7 Précautions

- Le panier créé est associé à une session **anonyme**, détruite à la fin de l'exécution avec le profil du navigateur. Selon l'application, ces paniers abandonnés peuvent toutefois s'accumuler côté serveur ou apparaître dans les statistiques. Signalez leur origine aux équipes métier.
- Sur une application réelle, identifiez le trafic de la sonde (chapitre 16, § 16.12) pour qu'il soit exclu des indicateurs commerciaux.

15.7.8 Erreurs courantes

Symptôme	Cause probable	Correction
Échec à « Vérifier le compteur du panier » alors que le message s'affiche	Ajout non enregistré : véritable anomalie fonctionnelle	Escalader vers l'équipe applicative (voir exemple 8)
Échec intermittent du compteur	Le compteur est mis à jour par un autre mécanisme (rechargement)	Vérifier directement le contenu du panier
Produit indisponible	Le produit de test a été retiré ou est en rupture	Convenir d'un produit de test permanent

15.8 Exemple 7 — Vérifier un parcours de commande en environnement de test

15.8.1 Objectif

Vérifier le tunnel de commande de bout en bout : connexion, panier, adresse, livraison, récapitulatif et confirmation. La commande de test créée est ensuite **annulée**. C'est le scénario le plus complet, et aussi le plus sensible, puisqu'il crée des données dans l'application.

> **Avertissement** — Un scénario qui passe commande **agit réellement** sur l'application. Ne l'exécutez **jamais** contre un environnement de production sans un accord formel des responsables métier et techniques, un moyen de paiement de test fourni par le prestataire de paiement, un produit et un compte de test identifiés, et une procédure d'annulation éprouvée. **Ne déclenchez jamais de paiement réel.** En cas de doute, limitez le scénario au récapitulatif, sans confirmation (§ 15.8.7).

15.8.2 Prérequis

- Environnement de **test** (recette) de l'application, identifiable de façon fiable.
- Paramètres `baseUrl`, `environnementAttendu` (`recette`), `autoriserCommande` (`oui`) et `produit`.
- Secrets `BOUTIQUE_USER` et `BOUTIQUE_PASSWORD` d'un compte de test dédié.
- Mode de paiement **simulé** sur l'environnement visé.
- Moyen d'**annuler** les commandes de test, dans le parcours ou par un traitement côté serveur convenu avec les équipes.

15.8.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/07-parcours-commande.mjs` :

JAVASCRIPT

```
// Exemple 7 — Vérifier un parcours de commande, EN ENVIRONNEMENT DE TEST UNIQUEMENT.
//
// Ce scénario crée une commande. Il ne doit JAMAIS être exécuté contre un site de
// production sans un accord formel du métier, un moyen de paiement de test et une
// procédure d'annulation. Deux garde-fous indépendants protègent son exécution :
//   1. le paramètre « autoriserCommande » doit valoir explicitement « oui » ;
//   2. la page doit déclarer l'environnement attendu (ici « recette ») ;
// faute de quoi le scénario s'arrête AVANT toute action irréversible.
// La commande de test créée est ensuite annulée (nettoyage).
//
// Paramètres : baseUrl, environnementAttendu (« recette »), autoriserCommande (« oui »), produit.
// Secrets : BOUTIQUE_USER, BOUTIQUE_PASSWORD (compte de test dédié).

export default async function scenario({ page, step, secrets, params, expect, log }) {
  const produit = params.produit || 'P100';
  const cliquerEtAttendre = (selecteur) =>
    Promise.all([page.waitForNavigation({ waitUntil: 'domcontentloaded' }),
    page.locator(selecteur).click()]);

  await step("Contrôler l'autorisation et l'environnement", async () => {
    // Garde-fou n° 1 : autorisation explicite portée par la configuration du contrôle.
    expect.equal(params.autoriserCommande, 'oui', 'Commande de test non autorisée par la
    configuration');
    await page.goto(`${params.baseUrl}/`, { waitUntil: 'domcontentloaded' });
    // Garde-fou n° 2 : l'application doit s'identifier comme environnement de test.
    const environnement = await page.$eval('meta[name=environnement]', (m) => m.content).catch(() =>
    null);
    expect.equal(environnement, params.environnementAttendu, "Environnement de l'application");
```

```
    await expect.visible(page, '[data-testid=env-banner]');
  });

  await step('Se connecter avec le compte de test', async () => {
    await page.goto(`${params.baseUrl}/connexion`, { waitUntil: 'domcontentloaded' });
    await page.locator('[data-testid=login-email]').fill(secrets.BOUTIQUE_USER);
    await page.locator('[data-testid=login-password]').fill(secrets.BOUTIQUE_PASSWORD);
    await cliquerEtAttendre('[data-testid=login-submit]');
    await expect.visible(page, '[data-testid=account-menu]');
  });

  await step('Constituer le panier', async () => {
    await page.goto(`${params.baseUrl}/produit/${produit}`, { waitUntil: 'domcontentloaded' });
    await Promise.all([
      page.waitForResponse((r) => r.url().includes('/api/panier') && r.request().method() ===
    'POST'),
      page.locator('[data-testid=add-to-cart]').click(),
    ]);
    await page.goto(`${params.baseUrl}/panier`, { waitUntil: 'domcontentloaded' });
    await expect.visible(page, `[data-testid=cart-line][data-product-id=${produit}]`);
  });

  await step('Renseigner l\'adresse', async () => {
    await cliquerEtAttendre('[data-testid=checkout-start]');
    await expect.visible(page, '[data-testid=checkout-address]');
    // Les champs sont préremplis par le compte de test ; on valide l'étape.
    await cliquerEtAttendre('[data-testid=checkout-next]');
  });

  await step('Choisir le mode de livraison', async () => {
    await expect.visible(page, '[data-testid=checkout-delivery]');
    await page.locator('[data-testid=delivery-pickup]').click();
    await cliquerEtAttendre('[data-testid=checkout-next]');
  });

  await step('Vérifier le récapitulatif', async () => {
    await expect.visible(page, '[data-testid=checkout-summary]');
    // Dernier contrôle avant l'action irréversible : le paiement doit être simulé.
    await expect.text(page, '[data-testid=payment-mode]', 'simulé');
  });

  let numero;
  await step('Confirmer la commande de test', async () => {
    await cliquerEtAttendre('[data-testid=order-confirm]');
    await expect.visible(page, '[data-testid=order-confirmation]');
    numero = await page.$eval('[data-testid=order-number]', (el) => el.textContent.trim());
    // Un numéro de commande de test doit respecter le format convenu.
    expect.ok(/^TEST-\d+$/.test(numero), `Numéro de commande inattendu : ${numero}`);
    log.info(`Commande de test créée : ${numero}`);
  });

  await step('Annuler la commande de test', async () => {
    // Nettoyage : la supervision ne doit pas laisser de commandes en attente de traitement.
    await cliquerEtAttendre('[data-testid=order-cancel]');
    await expect.text(page, '[data-testid=order-status]', 'annulee', { exact: true });
    log.info(`Commande de test annulée : ${numero}`);
  });
}
```

15.8.4 Explications

Deux garde-fous indépendants protègent l'exécution. Ils sont vérifiés **avant toute action** :

1. **Autorisation explicite** : le paramètre `autoriserCommande` doit valoir exactement « oui ». Un contrôle copié d'un autre environnement, ou mal configuré, s'arrête immédiatement.

2. **Identification de l'environnement** : l'application déclare son environnement dans une balise `<meta name="environnement">`. Le scénario compare cette valeur à `environnementAttendu`. Si l'adresse `baseUrl` désigne par erreur la production, la valeur lue ne correspond pas, et le scénario s'arrête sans rien créer.

Un troisième contrôle intervient juste avant l'action irréversible : l'étape « Vérifier le récapitulatif » s'assure que le mode de paiement affiché est **simulé**.

Nettoyage. La commande créée est annulée dans la dernière étape. Son numéro est journalisé, pour être retrouvé si l'annulation échoue.

> ! **Attention** — Ce nettoyage par le scénario **n'est pas garanti** : si l'exécution est interrompue entre la confirmation et l'annulation, la commande de test reste enregistrée, et aucune exécution suivante ne l'annulera.

Une interruption peut provenir d'un timeout, d'un redémarrage de la sonde, d'une coupure réseau ou d'un arrêt du navigateur ; chaque lancement crée sa propre commande. Pour ce parcours :

- prévoyez un **nettoyage indépendant côté environnement de recette**, convenu avec ses responsables : par exemple, une purge planifiée des commandes dont le numéro commence par `TEST-` et qui ont été créées par le compte de test dédié, quel que soit leur statut ;
- configurez le contrôle **sans nouvelle tentative** (`"retry": { "attempts": 0 }`, chapitre 8, § 8.5) : après un échec ambigu, par exemple un timeout survenu après la confirmation, une nouvelle tentative créerait une seconde commande, alors que la première a peut-être été enregistrée : le numéro étant attribué par l'application à chaque création, rien ne lui permet de reconnaître la répétition (chapitre 8, § 8.5.2) ;
- surveillez le nombre de commandes `TEST-…` non annulées (ligne « Commandes de test non annulées » du § 15.8.8).

> ✓ **Conseil** — Demandez aux équipes de développement d'exposer l'environnement de manière fiable : balise `meta`, en-tête HTTP ou point d'information dédié. Un garde-fou fondé sur l'adresse seule est fragile : une adresse de recette peut être redirigée, ou recopiée par erreur dans un contrôle de production.

15.8.5 Résultat attendu

État `OK`, huit étapes. Le journal contient le numéro de la commande de test créée, puis annulée :

```json
"log": [
  { "niveau": "INFO", "message": "Commande de test créée : TEST-00001" },
  { "niveau": "INFO", "message": "Commande de test annulée : TEST-00001" }
]
```

Si l'environnement ne correspond pas, ici avec `environnementAttendu=production`, le scénario s'arrête à la première étape, **sans aucune action sur l'application** :

```json
{
  "status": "CRITICAL",
  "error": {
    "type": "assertion_failed",
    "step": "Contrôler l'autorisation et l'environnement",
    "message": "Environnement de l'application : attendu « production », obtenu « recette »"
  }
}
```

15.8.6 Ordonnancement

Ce scénario est long et sollicite plusieurs services. Planifiez-le à une fréquence modérée, par exemple toutes les 15 à 30 minutes, et de préférence dans un plan d'exécution qui commence par les scénarios de connexion et d'ajout au panier avec l'option `stopOnFailure` (chapitre 12). Si la connexion échoue, il est inutile de tenter la commande : l'alerte porte alors directement sur la cause.

15.8.7 Variantes

- **Parcours sans confirmation** : arrêtez le scénario après « Vérifier le récapitulatif ». Le tunnel est vérifié jusqu'au paiement, sans créer de commande. C'est la variante recommandée lorsqu'aucun environnement de test n'est disponible.
- **Paiement de test d'un prestataire** : utilisez les moyens de paiement de test documentés par votre prestataire, exclusivement sur l'environnement de test associé.

15.8.8 Erreurs courantes

Symptôme	Cause probable	Correction
Échec au premier contrôle	Paramètres absents ou incorrects ; balise d'environnement absente	Vérifier la configuration ; convenir d'un marqueur d'environnement
Échec à « Annuler la commande de test »	Bouton d'annulation absent pour ce statut de commande	Annuler manuellement la commande journalisée ; prévenir les équipes métier
Commandes de test non annulées qui s'accumulent	Scénario interrompu entre la confirmation et l'annulation (timeout, incident)	Mettre en place une purge côté serveur des commandes `TEST-…` ; surveiller leur nombre
Le scénario dépasse son timeout	Parcours long sur un Raspberry Pi 5	Porter le timeout du contrôle à deux à trois fois la durée normale (chapitre 8, § 8.4)

15.9 Exemple 8 — Détecter une erreur fonctionnelle

15.9.1 Objectif

Détecter une anomalie **invisible pour une supervision technique** : toutes les réponses HTTP valent 200, la page affiche « Produit ajouté au panier », et pourtant le résultat métier est faux. Le panier peut être vide, ou son total incohérent. Le scénario **recalcule** le résultat attendu et le compare à ce que l'application affiche. Il vérifie aussi l'absence d'erreurs JavaScript et de réponses serveur en erreur pendant le parcours.

15.9.2 Prérequis

- Paramètres `baseUrl`, `produit` (par défaut `P300`) et `quantite` (par défaut 2).
- Paramètre `panne` : **réservé à la démonstration**, il active une panne simulée de l'application fictive. Il n'a pas d'équivalent dans une application réelle.

15.9.3 Scénario scripté

Fichier `exemples/puppeteer/scenarios/08-detection-erreur-fonctionnelle.mjs` :

```javascript
// Exemple 8 — Détecter une erreur fonctionnelle invisible pour une supervision technique.
//
// Toutes les réponses HTTP de ce parcours valent 200, et l'application affiche
// « Produit ajouté au panier ». Pourtant, le résultat métier peut être faux :
// panier vide, ou total incohérent. Ce scénario recalcule le résultat attendu et
```

```
// le compare à ce que l'application affiche.
//
// Pour la démonstration, l'application fictive simule deux pannes :
//   --param panne=panier  : l'ajout est confirmé mais rien n'est enregistré ;
//   --param panne=prix    : le total du panier ignore la quantité.
// Sans paramètre « panne », le scénario doit réussir.
//
// Paramètres : baseUrl, produit (défaut P300), quantite (défaut 2), panne (démonstration).

export default async function scenario({ page, step, params, expect, log }) {
  const produit = params.produit || 'P300';
  const quantite = Number(params.quantite || 2);
  const suffixe = params.panne ? `?panne=${encodeURIComponent(params.panne)}` : '';

  // Indices techniques collectés pendant tout le parcours.
  const erreursJavaScript = [];
  const reponsesEnErreur = [];
  page.on('pageerror', (e) => erreursJavaScript.push(e.message));
  page.on('response', (r) => { if (r.status() >= 500) reponsesEnErreur.push(`${r.status()}
    ${r.url()}`); });

  let prixUnitaire;

  await step('Ouvrir la fiche produit', async () => {
    await page.goto(`${params.baseUrl}/produit/${produit}${suffixe}`, { waitUntil: 'domcontentloaded'
    });
    // Le prix de référence est lu dans un attribut technique stable, pas dans le texte formaté.
    prixUnitaire = Number(await page.$eval('[data-testid=product-price]', (el) => el.dataset.value));
    expect.ok(prixUnitaire > 0, `Prix unitaire invalide : ${prixUnitaire}`);
  });

  // Nom d'étape littéral et stable : il ne doit pas dépendre d'une valeur variable.
  await step('Ajouter le produit au panier', async () => {
    // fill() remplace le contenu existant du champ quantité.
    await page.locator('[data-testid=quantity]').fill(String(quantite));
    await Promise.all([
      page.waitForResponse((r) => r.url().includes('/api/panier') && r.request().method() ===
    'POST'),
      page.locator('[data-testid=add-to-cart]').click(),
    ]);
    await expect.text(page, '[data-testid=add-to-cart-message]', 'ajouté');
  });

  await step('Vérifier le panier', async () => {
    await page.goto(`${params.baseUrl}/panier${suffixe}`, { waitUntil: 'domcontentloaded' });
    // Le message de succès ne suffit pas : la ligne doit exister avec la bonne quantité.
    await expect.visible(page, `[data-testid=cart-line][data-product-id=${produit}]`);
    const quantiteAffichee = Number(
      await page.$eval(`[data-testid=cart-line][data-product-id=${produit}] [data-testid=cart-line-
    qty]`, (el) => el.textContent),
    );
    expect.equal(quantiteAffichee, quantite, 'Quantité dans le panier');
  });

  await step('Vérifier la cohérence du total', async () => {
    const totalAffiche = Number(await page.$eval('[data-testid=cart-total]', (el) =>
    el.dataset.value));
    const totalAttendu = Math.round(prixUnitaire * quantite * 100) / 100;
    log.info(`Total attendu ${totalAttendu}, total affiché ${totalAffiche}`);
    expect.equal(totalAffiche, totalAttendu, 'Total du panier');
  });

  await step('Vérifier l\'absence d\'erreurs techniques', async () => {
    expect.ok(erreursJavaScript.length === 0, `Erreurs JavaScript : ${erreursJavaScript.join(' |
    ')}`);
    expect.ok(reponsesEnErreur.length === 0, `Réponses serveur en erreur : ${reponsesEnErreur.join('
    | ')}`);
```

```
  });
}
```

15.9.4 Explications

- **Valeur de référence.** Le prix unitaire est lu dans un attribut technique (`data-value`), et non dans le texte affiché (« 89,50 € »), dont le format dépend de la langue.
- **Calcul indépendant.** Le total attendu est calculé par le scénario (prix × quantité), puis comparé au total affiché. C'est cette comparaison qui révèle l'anomalie.
- **Indices techniques.** Le scénario s'abonne, avant toute navigation, aux événements `pageerror` (erreurs JavaScript non gérées) et `response` (réponses HTTP). Il vérifie en dernière étape qu'aucune erreur n'a été collectée. Le moteur de scénarios collecte aussi ces informations pour le diagnostic. Ici, le scénario en fait un **critère d'échec** explicite.
- **Nom d'étape stable.** L'étape s'appelle « Ajouter le produit au panier », et non « Ajouter 2 produits » : la quantité est un paramètre, et le nom ne doit pas varier (chapitre 14, § 14.5.4).

15.9.5 Résultats attendus

Sans panne simulée, l'état est `OK`, avec cinq étapes réussies.

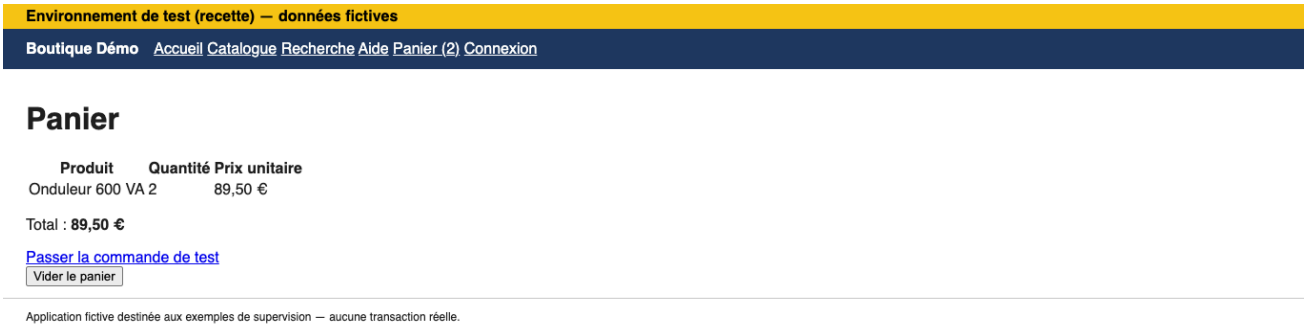


Figure 15-2 — Capture produite par l'exemple 8 avec la panne « prix » de la Boutique Démo : le total du panier est incohérent

Avec la panne `prix` (`--param panne=prix`), le total ignore la quantité. Le scénario échoue à la bonne étape, avec un message qui donne immédiatement la nature du problème. Résultat obtenu avec le lanceur local :

JSON


```
{
  "checkId": "08-detection-erreur-fonctionnelle",
  "status": "CRITICAL",
  "durationMs": 600,
  "error": {
    "type": "assertion_failed",
    "step": "Vérifier la cohérence du total",
    "message": "Total du panier : attendu « 179 », obtenu « 89.5 »"
  },
  "steps": [
    { "index": 0, "name": "Ouvrir la fiche produit", "status": "OK", "durationMs": 74 },
    { "index": 1, "name": "Ajouter le produit au panier", "status": "OK", "durationMs": 507 },
    { "index": 2, "name": "Vérifier le panier", "status": "OK", "durationMs": 17 },
    { "index": 3, "name": "Vérifier la cohérence du total", "status": "CRITICAL", "durationMs": 2 }
  ],
  "artifacts": [
    { "name": "failure.png", "kind": "screenshot", "bytes": 39086 },
    { "name": "console.json", "kind": "console", "bytes": 226 }
  ]
}
```


Avec la panne `panier` (`--param panne=panier`), l'application confirme l'ajout mais n'enregistre rien. Le scénario échoue à l'étape « Vérifier le panier » :

JSON


```
{
  "status": "CRITICAL",
  "error": {
    "type": "assertion_failed",
    "step": "Vérifier le panier",
    "message": "Élément [data-testid=cart-line][data-product-id=P300] non visible après 10000 ms"
  }
}
```

Dans les deux cas, toutes les réponses HTTP valaient 200 : aucun contrôle technique n'aurait détecté ces anomalies.

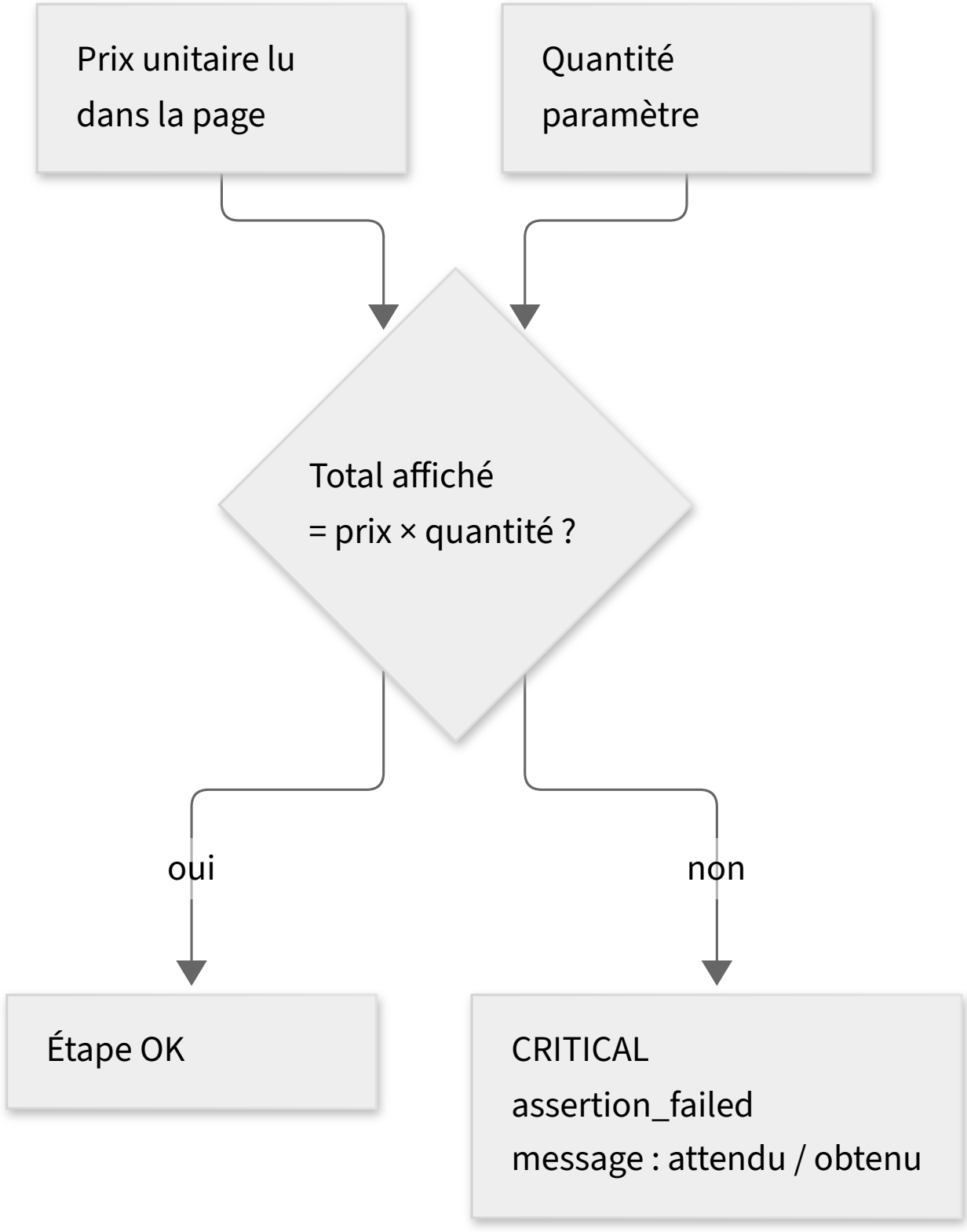


Schéma 15-1 — Vérification de cohérence du total du panier

15.9.6 Variantes

- **Cohérence entre pages** : comparez le total du panier au montant du récapitulatif de commande.

- **Règles métier** : frais de port offerts au-delà d'un montant, remise appliquée pour un code promotionnel de test, taxes correctement calculées.
- **Indicateurs métier** : pour des règles portant sur des volumes, par exemple le nombre de commandes de la journée, préférez un contrôle de base de données (chapitre 11), plus léger qu'un scénario navigateur.

15.9.7 Erreurs courantes

Symptôme	Cause probable	Correction
Échec sur l'arrondi	Comparaison de nombres décimaux sans arrondi	Arrondir au centime avant comparaison (comme dans l'exemple)
Échec à « Vérifier l'absence d'erreurs techniques » sans impact visible	Erreur JavaScript d'un script tiers (mesure d'audience)	Bloquer le domaine tiers (`blockDomains`) ou filtrer les erreurs par origine
Différence de format	Lecture d'un texte formaté au lieu d'une valeur technique	Lire un attribut `data-*` convenu avec les développeurs

15.10 Synthèse des essais

Les exemples ont été exécutés sur poste de travail avec le lanceur local, Puppeteer 25.18.1 et Chrome for Testing 157, contre l'application de démonstration. Ces essais portent sur l'environnement du poste de travail ; sur la sonde, qui utilise le Chromium des dépôts Raspberry Pi OS, chaque scénario se vérifie par une exécution d'essai (chapitre 14, § 14.16). Les cas d'échec ont été provoqués volontairement pour vérifier que chaque scénario échoue à la bonne étape, avec un message exploitable.

Exécution	État obtenu	Étape en échec	Type d'erreur
Exemples 1 à 8, versions scriptées	`OK`	—	—
Exemples 1 à 6, versions déclaratives (dont 3 bis)	`OK`	—	—
Exemple 5, terme sans résultat	`CRITICAL`	Vérifier les résultats	`assertion_failed`
Exemple 3, mot de passe erroné	`CRITICAL`	Vérifier l'accès à l'espace client	`assertion_failed`
Exemple 3, secret absent	`UNKNOWN`	Avant la première étape	`secret_missing`
Exemple 7, environnement attendu « production »	`CRITICAL`	Contrôler l'autorisation et l'environnement	`assertion_failed`
Exemple 8, panne `prix`	`CRITICAL`	Vérifier la cohérence du total	`assertion_failed`
Exemple 8, panne `panier`	`CRITICAL`	Vérifier le panier	`assertion_failed`
Exemple 4, timeout global de 300 ms	`CRITICAL`	Étape en cours à l'expiration	`timeout`
Exemple 1, application arrêtée	`CRITICAL`	Ouvrir la page d'accueil	`connection_refused`

15.11 Adapter les exemples à vos applications

Pour transformer un exemple en scénario de supervision de votre application :

1. **Formalisez l'intention** : ce que le scénario doit prouver, l'étape la plus critique, le résultat attendu, le compte de test et les données de test.
2. **Choisissez le mode** : déclaratif si le parcours est linéaire, scripté sinon (chapitre 14, § 14.1.1).

3. **Identifiez les sélecteurs** dans les outils de développement de Chrome. Demandez des attributs `data-testid` pour les éléments critiques (chapitre 16, § 16.1).
4. **Remplacez les valeurs propres à l'environnement** par des paramètres (`baseUrl` , produit, terme) et les identifiants par des secrets.
5. **Ajoutez des assertions** à chaque étape importante, avec des messages explicites.
6. **Testez sur poste de travail**, en succès **et** en échec.
7. **Déployez** sur la sonde, désactivé, puis lancez une exécution d'essai (chapitre 14, § 14.16).
8. **Définissez la planification et les seuils** : fréquence, timeout, nouvelles tentatives, `warningDuration` .
9. **Documentez** le scénario : intention, responsables de l'application, compte de test, procédure en cas d'alerte.

Le chapitre 16 rassemble les bonnes pratiques qui garantissent la fiabilité de ces scénarios dans la durée.

Chapitre 16 — Bonnes pratiques des scénarios

Un scénario de supervision n'a de valeur que s'il est **fiable** : il doit échouer lorsque le service est dégradé, et seulement dans ce cas. Un scénario qui produit de fausses alertes finit par être ignoré, et un scénario qui ne détecte rien donne une fausse assurance. Ce chapitre rassemble les pratiques qui garantissent cette fiabilité dans la durée, sur le matériel contraint qu'est un Raspberry Pi 5, et dans le respect de la sécurité des applications supervisées.

Chaque section se termine par une courte liste de règles. Le § 16.17 les reprend sous forme de liste de contrôle, à appliquer avant toute mise en service.

16.1 Utiliser des sélecteurs robustes

Les sélecteurs sont la première cause de **faux positifs**, c'est-à-dire d'alertes déclenchées alors que l'application fonctionne. Lorsqu'une page évolue (nouvelle mise en forme, réorganisation d'un formulaire, changement de libellé), un sélecteur fragile cesse de correspondre et le scénario échoue.

L'ordre de préférence est le suivant (chapitre 14, § 14.7.1) :

1. **attribut dédié aux tests** : `[data-testid=add-to-cart]` ;
2. **identifiant stable** : `#login-submit`, s'il n'est pas généré automatiquement ;
3. **nom d'accessibilité** : `::-p-aria(Ajouter au panier)` ;
4. **texte visible** : `::-p-text(Ajouter au panier)`, en sachant qu'il dépend des libellés et de la langue ;
5. **attribut fonctionnel** : `input[name=email]` ;
6. **XPath relatif**, en dernier recours.

À proscrire :

- les sélecteurs **positionnels** (`div:nth-child(3) > ul > li > a`) et les XPath **absolus** (`/html/body/div[2]/table[1]/tbody/tr[3]/td[1]/a`) ;
- les **classes de style** (`.btn-blue`), qui changent à chaque refonte graphique ;
- les **identifiants générés** par les frameworks (`#ember1234`, `#mui-17`), qui changent d'une version à l'autre, voire d'un chargement à l'autre.

> **Information** — La génération précédente recommandait d'utiliser systématiquement des chemins XPath obtenus par « Copy XPath ». Cette recommandation est abandonnée : ces chemins absolus décrivent la position d'un élément dans la page, et cassent à la moindre évolution de la structure.

Un bon sélecteur est aussi **unique**. S'il correspond à plusieurs éléments, l'action porte sur le premier, qui n'est pas forcément celui que vous visez. Vérifiez-le dans la console des outils de développement :

```javascript
// Dans la console de Chrome : nombre d'éléments correspondant au sélecteur
document.querySelectorAll('[data-testid=add-to-cart]').length
```

Règles

- Privilégier `data-testid` et le demander aux équipes de développement (§ 16.16).
- Ne jamais utiliser de sélecteur positionnel ni de XPath absolu.

- Vérifier l'unicité de chaque sélecteur d'action.

16.2 Écrire des assertions explicites

Une action réussie ne prouve pas que le service fonctionne. Un clic sur « Se connecter » réussit même lorsque l'authentification échoue. C'est l'**assertion** qui suit qui fait du scénario un contrôle.

Principes :

- **une assertion à chaque étape importante**, qui vérifie l'effet attendu de l'étape ;
- **vérifier le résultat métier**, pas l'accusé de réception : le contenu du panier plutôt que le message « Produit ajouté » (chapitre 15, exemple 8) ;
- **des messages qui permettent le diagnostic** : « Total du panier : attendu 179, obtenu 89,5 » plutôt que « Erreur » ;
- **comparer des valeurs techniques** (attributs `data-*`) plutôt que des textes formatés, qui dépendent de la langue, de la devise et de la mise en forme ;
- **tolérer ce qui varie légitimement** : un nombre minimal de résultats plutôt qu'un nombre exact, une expression régulière plutôt qu'un texte complet.

JAVASCRIPT

```
// Faible : prouve seulement que le bouton existait
await page.locator('[data-testid=add-to-cart]').click();

// Robuste : prouve que le produit est réellement dans le panier
await page.goto(`${params.baseUrl}/panier`, { waitUntil: 'domcontentloaded' });
await expect.visible(page, `[data-testid=cart-line][data-product-id=${produit}]`);
```

Règles

- Terminer chaque étape importante par une assertion.
- Vérifier le résultat métier, pas seulement l'affichage d'un message.
- Rédiger des messages d'échec qui indiquent la valeur attendue et la valeur obtenue.

16.3 Maîtriser les délais

Trois niveaux de délais s'appliquent : le timeout global du contrôle, le délai de navigation et le délai d'attente des éléments (chapitre 14, § 14.10.5). Les réglages recommandés sont les suivants.

Réglage	Recommandation
Timeout du contrôle (`timeout`)	Deux fois la durée normale du parcours, et au moins 30 secondes
Délai d'une attente particulière	Laisser la valeur par défaut. L'augmenter seulement pour une opération connue pour être lente, et seulement pour cette attente
Seuil d'avertissement (`thresholds.warningDuration`)	Environ 1,5 fois la durée habituelle, mesurée sur plusieurs jours
Nouvelles tentatives (`retry`)	Une tentative après 30 à 60 secondes pour les parcours importants

Le délai ne doit pas servir à **masquer une lenteur**. Un parcours qui prend habituellement 6 secondes et qui en prend soudain 25 révèle une dégradation. Avec un timeout de 60 secondes et un seuil d'avertissement à 9 secondes, cette dégradation est signalée par l'état `WARNING`, sans fausse alerte `CRITICAL`.

Pensez aussi aux performances de la sonde. Un Raspberry Pi 5 exécute un navigateur plus lentement qu'un poste de travail récent. Mesurez la durée des scénarios **sur la sonde**, avec des exécutions d'essai, avant de fixer les seuils.

Règles

- Fixer les seuils à partir de durées mesurées sur la sonde, pas sur le poste de travail.
- Signaler la lenteur par `warningDuration`, et réserver l'échec aux dysfonctionnements.
- Ne jamais allonger les délais globalement pour faire disparaître des échecs intermittents sans en comprendre la cause.

16.4 Attendre des événements plutôt que temporiser

Une pause fixe est la cause la plus fréquente des scénarios instables (chapitre 14, § 14.10.1). Remplacez-la systématiquement par l'attente de l'événement qui marque la fin de l'opération :

Situation	Au lieu d'une pause, attendre…
Après un clic qui charge une page	La navigation, armée avant le clic (`Promise.all`)
Après un clic qui affiche un contenu	L'élément de ce contenu (`waitForSelector`, `expect.visible`)
Après une action qui appelle une API	La réponse de cette API (`waitForResponse`)
Pendant un indicateur de chargement	La disparition de l'indicateur (`{ hidden: true }`)
Après une mise à jour d'un compteur	La condition (`waitForFunction`)

Le mode déclaratif applique ce principe automatiquement : chaque action attend son élément, et chaque assertion attend sa condition.

Règles

- Aucune pause fixe dans un scénario.
- Armer les attentes de navigation et de réponse **avant** l'action qui les déclenche.

16.5 Gérer les erreurs sans les masquer

Le moteur de scénarios intercepte les exceptions, marque l'étape en échec, collecte les éléments de diagnostic et détermine l'état du contrôle (chapitre 14, § 14.12). Le script n'a donc presque jamais besoin de `try/catch`.

- **Ne jamais intercepter une erreur pour l'ignorer.** Un échec masqué devient un succès apparent, ce qui est le pire résultat pour un contrôle de supervision.
- **Intercepter uniquement l'absence normale d'un élément facultatif**, comme un bandeau de consentement ou une fenêtre promotionnelle (chapitre 14, § 14.8.6), avec un délai court.
- **Enrichir un message** si nécessaire, en levant une nouvelle erreur plus explicite.
- **Tester le cas d'échec** de chaque scénario avant sa mise en service (chapitre 14, § 14.15.3).

Règles

- Laisser remonter les exceptions.
- N'intercepter que les éléments facultatifs, avec un délai court.
- Vérifier que le scénario échoue à la bonne étape lorsque l'application est défaillante.

16.6 Limiter la consommation de mémoire et de processeur

Sur un Raspberry Pi 5, le navigateur est le composant le plus coûteux de la sonde. La mémoire disponible détermine le nombre de scénarios exécutables simultanément (un par défaut) et la stabilité de l'ensemble. Les pratiques suivantes réduisent la consommation de chaque exécution :

Pratique	Effet
Bloquer les domaines tiers inutiles à la vérification (`blockDomains`) : mesure d'audience, publicité, widgets sociaux, cartes	Moins de requêtes, moins de scripts exécutés, moins de mémoire ; parcours plus rapide et plus stable (§ 16.11)
Garder la taille de fenêtre par défaut (`viewport`) ; ne pas simuler un écran 4K	Moins de mémoire pour le rendu et les captures
Limiter le nombre de pages parcourues à ce qui est nécessaire pour l'intention du scénario	Exécution plus courte
Fermer les fenêtres secondaires ouvertes par le scénario dès qu'elles ne sont plus utiles	Libère la mémoire pendant l'exécution
Ne pas accumuler de grandes quantités de données dans le script (contenus de pages entiers, listes de milliers d'éléments)	Mémoire du processus Node.js
Espacer les exécutions : toutes les 5 à 15 minutes plutôt que toutes les minutes	Moins de charge cumulée, plus de marge pour les nouvelles tentatives
Regrouper les scénarios d'une même application dans un plan d'exécution	Évite les exécutions concurrentes et les échéances qui se chevauchent (chapitre 12)

Surveillez les indicateurs de santé de la sonde : mémoire, température, retard du planificateur (chapitre 17). Un retard croissant signale que la capacité d'exécution est saturée.

Règles

- Bloquer les ressources tierces non nécessaires.
- Espacer les exécutions des scénarios.
- Surveiller le retard du planificateur et la mémoire de la sonde.

16.7 Gérer les sessions et les comptes de test

Chaque exécution démarre avec un profil vierge (chapitre 13, § 13.3.3). Chaque scénario qui a besoin d'une session doit donc s'authentifier. Ce choix est délibéré : la connexion fait partie du service vérifié, et aucune session ne subsiste sur la sonde.

Le **compte de test** utilisé doit être :

- **dédié** à la supervision, nommé de façon explicite (exemple fictif : `test-supervision@example.com`) et jamais partagé avec des personnes ;
- **limité en droits** au strict nécessaire au parcours : pas de droits d'administration, pas d'accès à des données réelles de clients ;
- **exempté du second facteur** si nécessaire, uniquement avec l'accord de l'équipe sécurité et des mesures compensatoires (restriction aux adresses de la sonde, mot de passe long) ;
- **connu des équipes** : support, sécurité et métier doivent savoir que ses connexions régulières sont normales ;
- **protégé contre le verrouillage** : une politique de verrouillage après plusieurs échecs peut bloquer la supervision. Convenez d'un seuil adapté et limitez les nouvelles tentatives.

Lorsque plusieurs sondes supervisent la même application depuis des sites différents, prévoyez un compte de test **par sonde**. Une connexion de l'une ne fermera pas la session de l'autre, et l'origine d'une anomalie sera identifiable.

Règles

- Un compte de test dédié, aux droits minimaux, connu des équipes.
- Une politique de verrouillage et des nouvelles tentatives compatibles.
- Un compte par sonde lorsque plusieurs sondes visent la même application.

16.8 Isoler les scénarios

Un scénario doit pouvoir s'exécuter **seul**, à tout moment, quel que soit le résultat des autres scénarios et des exécutions précédentes.

- **Pas de dépendance à l'état laissé par un autre scénario.** Un scénario de commande ne doit pas supposer que le scénario d'ajout au panier a rempli le panier : il constitue lui-même son panier (chapitre 15, exemple 7).
- **Pas de dépendance à l'ordre d'exécution.** Les plans d'exécution enchaînent des scénarios pour éviter des tests inutiles (`stopOnFailure`), mais chaque scénario reste autonome (chapitre 12).
- **Des données de test indépendantes.** Si deux scénarios modifient des données, ils utilisent des données distinctes (produit, compte), afin de ne pas interférer.
- **Pas d'état stocké entre deux exécutions.** Le moteur ne conserve rien d'une exécution à l'autre, et le script ne doit pas chercher à contourner ce principe.

Règles

- Chaque scénario constitue lui-même les données dont il a besoin.
- Aucun scénario ne dépend du résultat d'un autre.

16.9 Nettoyer les ressources et les données

Le moteur de scénarios nettoie tout ce qui concerne la sonde : navigateur, processus, profil, cache (chapitre 14, § 14.14). Le script n'a pas à s'en préoccuper.

En revanche, les **données créées dans l'application supervisée** relèvent de la responsabilité du scénario et des équipes qui l'exploitent :

- **annuler ou supprimer** ce que le scénario crée (commande de test, réservation, message), dans le scénario lui-même lorsque c'est possible (chapitre 15, exemple 7) ;
- **journaliser l'identifiant** de toute donnée créée, pour pouvoir la retrouver si le nettoyage échoue ;
- **prévoir un nettoyage côté serveur** des données de test résiduelles, par exemple une purge périodique des commandes `TEST-…` mise en place par l'équipe applicative ;
- **préférer les parcours sans effet de bord** lorsqu'ils suffisent : arrêter le tunnel de commande au récapitulatif vérifie l'essentiel sans rien créer.

Règles

- Ne jamais fermer le navigateur dans le script : le moteur s'en charge.
- Annuler ce que le scénario crée, et journaliser les identifiants créés.
- Prévoir une purge côté serveur des données de test.

16.10 Protéger les secrets

Les identifiants des comptes de test donnent accès à vos applications. Ils sont traités comme des secrets (chapitre 9) :

- **jamais dans le code** d'un script ni dans un scénario déclaratif : toujours `secrets.NOM` ou `${secret:NOM}` ;
- **jamais dans les paramètres** (`params`), qui sont visibles par tous les utilisateurs pouvant consulter le contrôle ;
- **jamais dans les journaux** : n'écrivez pas `log.info(secrets.BOUTIQUE_PASSWORD)`. La sonde masque les valeurs connues des secrets, mais ce masquage est une protection de dernier recours, pas une permission ;
- **jamais dans un message d'assertion** : ne comparez pas un texte de la page à un secret ;
- **déclarés au plus juste** : un contrôle ne déclare que les secrets dont il a besoin. Le moteur ne lui transmet que ceux-là ;
- **renouvelés régulièrement**, selon la politique de l'entreprise, et immédiatement en cas de doute. Le remplacement d'un secret sur la sonde ne nécessite aucune modification des scénarios.

Sur poste de travail, le lanceur local lit les secrets dans des variables d'environnement. Ne les écrivez pas dans des fichiers versionnés, et ne les passez pas en clair sur la ligne de commande.

Règles

- Aucune valeur sensible dans le code, les paramètres ou les journaux.
- Déclarer uniquement les secrets nécessaires.
- Renouveler les secrets selon la politique de l'entreprise.

16.11 Maîtriser les ressources tierces

Une page moderne charge de nombreuses ressources de **domaines tiers** : mesure d'audience, publicité, réseaux sociaux, discussion en ligne, tests A/B. Elles sont rarement utiles à la vérification du parcours et posent trois problèmes :

- elles ralentissent le scénario et faussent la mesure de performance de **votre** application ;
- elles consomment de la mémoire sur la sonde ;
- leurs propres défaillances (script en erreur, service lent) peuvent faire échouer un scénario alors que votre application fonctionne.

La propriété `blockDomains` du contrôle bloque les requêtes vers les domaines indiqués et leurs sous-domaines :

```json
"browser": {
  "mode": "script",
  "scriptRef": "boutique-panier@sha256:9a7e31c4d2b8",
  "blockDomains": ["stats.example.net", "ads.example.org", "chat.example.com"]
}
```

À l'inverse, **ne bloquez pas** les services tiers qui font partie du service rendu : fournisseur d'identité, module de paiement, réseau de diffusion des ressources de l'application, cartographie d'une page de recherche de magasins. Leur défaillance est une défaillance réelle du service, que le scénario doit détecter.

Règles

- Bloquer les domaines de mesure d'audience, de publicité et de widgets non essentiels.
- Ne jamais bloquer un service tiers qui participe au parcours vérifié.

16.12 Identifier le trafic de la sonde

Le trafic de la sonde doit être reconnaissable par les équipes de l'application, pour trois raisons :

- **statistiques** : les exécutions régulières ne doivent pas fausser l'audience ni les indicateurs commerciaux ;
- **sécurité** : les connexions répétées du compte de test ne doivent pas être prises pour une attaque ;
- **diagnostic** : les journaux de l'application doivent permettre de retrouver les requêtes d'une exécution.

Moyens disponibles :

Moyen	Mise en œuvre
Suffixe d'identification du navigateur	Propriété `userAgentSuffix` du contrôle (par exemple `SupervisionSonde`)
Adresse IP de la sonde	Communiquée aux équipes ; adresse fixe recommandée (chapitre 7)
Compte de test dédié	Exclu des statistiques par son identifiant
Refus des cookies non essentiels	Le bandeau de consentement est refusé par le scénario (chapitre 14, § 14.8.6)

Règles

- Rendre le trafic de la sonde identifiable, et en informer les équipes de l'application.
- Faire exclure ce trafic des statistiques d'audience et des indicateurs métier.

16.13 Captures d'écran, traces et données personnelles

En cas d'échec, la sonde enregistre automatiquement une capture d'écran et les éléments de diagnostic (console, requêtes en erreur). Ces **artefacts** sont précieux pour comprendre un incident (chapitre 17). Ils peuvent aussi contenir des informations sensibles.

- **Ne réalisez pas de capture manuelle** dans les scripts : la capture automatique en cas d'échec suffit, et elle est gérée avec une rétention et un quota (chapitre 20).
- **Utilisez des données fictives.** Avec un compte de test sans données réelles, les captures ne montrent aucune donnée personnelle.
- **Évitez les pages sensibles** lorsqu'elles ne sont pas nécessaires au parcours : données bancaires, documents d'identité, informations de santé.
- **Limitez l'accès** aux résultats détaillés aux personnes qui en ont besoin, à l'aide des rôles (chapitre 9).
- **Conservez les artefacts le temps nécessaire**, sans plus. La rétention par défaut est courte (7 jours).

> ! **Attention** — Les champs de mot de passe sont masqués à l'écran par le navigateur, et donc sur les captures. En revanche, une page peut afficher en clair une donnée personnelle, comme une adresse ou un numéro de client. Un compte de test sans données réelles est la meilleure protection.

Les mesures ci-dessus réduisent le risque sans le supprimer. Gardez à l'esprit les limites suivantes :

Élément	Ce qui est protégé	Ce qui ne l'est pas
Masquage automatique	Les **valeurs connues** des secrets déclarés par le contrôle sont remplacées dans les journaux, la console du navigateur et les messages d'erreur	Une donnée personnelle affichée par la page ; un secret **réaffiché** par l'application sous une autre forme (tronqué, encodé, reformaté), ou un secret qui n'est pas déclaré dans le contrôle. Une capture d'écran n'est jamais masquée
Capture d'écran automatique	Elle n'est produite qu'en cas d'échec	Elle montre la page telle qu'elle était affichée au moment de l'échec, données comprises

Élément	Ce qui est protégé	Ce qui ne l'est pas
Journal de console	Messages d'erreur de la page, secrets connus masqués	Les messages écrits par l'application elle-même peuvent contenir des identifiants, des adresses électroniques ou des jetons de session
Rôles	Seuls les rôles autorisés consultent et téléchargent les artefacts (chapitre 9)	Un artefact téléchargé, puis copié dans un ticket ou une messagerie, échappe à la rétention et aux rôles de la sonde

En conséquence :

- **Désactivez la capture** pour un contrôle qui traverse des pages sensibles qu'il ne peut pas éviter : `"captureOnFailure": false` dans la section `browser` (chapitre 14, § 14.2). Le diagnostic s'appuie alors sur l'étape en échec, le message d'erreur et le journal de console ; compensez par des noms d'étapes et des messages d'assertion précis (§ 16.2).
- **N'écrivez pas de donnée de la page dans les messages** du script (`log.info`, messages d'assertion) lorsqu'elle peut être personnelle : préférez un comptage ou un indicateur (« 3 lignes dans le panier ») à la valeur elle-même.
- **Respectez la rétention** : ne conservez pas de copie locale des artefacts au-delà de l'analyse de l'incident.
- **Contrôlez tout artefact avant de le transmettre**, en particulier au support : ouvrez la capture et le journal, puis retirez ou masquez tout élément sensible. Il en va de même pour l'archive de diagnostic, qui contient des journaux et peut contenir des captures (chapitre 19, § 19.12). Le canal de transfert vers le support et la durée de conservation par l'éditeur sont définis dans sa politique de support, selon les conditions de votre contrat, et indiqués sur le portail du support (`https://support-synovex.incitypark.fr`).

Règles

- Pas de capture manuelle dans les scripts.
- Comptes et données de test exclusivement.
- Accès aux artefacts limité par les rôles ; rétention courte.
- Capture désactivée (`captureOnFailure: false`) sur les parcours qui traversent des pages sensibles.
- Contrôle du contenu de tout artefact ou archive avant transmission.

16.14 Fiabilité dans la durée

Un scénario est fiable lorsqu'il produit le même résultat dans les mêmes conditions. Les échecs **intermittents**, qui disparaissent à l'exécution suivante sans intervention, minent la confiance des équipes. Ils doivent être traités comme de vrais défauts du scénario ou de l'application.

Cause d'instabilité	Remède
Pause fixe ou attente mal armée	Attendre l'événement adéquat, armé avant l'action (§ 16.4)
Élément facultatif (bandeau, fenêtre promotionnelle) qui recouvre un bouton	Fermer l'élément s'il est présent (chapitre 14, § 14.8.6)
Données de test qui changent (produit retiré, terme sans résultat)	Données de référence convenues et pérennes
Service tiers instable	Bloquer s'il n'est pas essentiel ; sinon, l'échec est légitime
Lenteur ponctuelle du réseau	Une nouvelle tentative (`retry.attempts: 1`)
Sonde saturée	Espacer les exécutions, regrouper en plans d'exécution (§ 16.6)

Les **nouvelles tentatives** absorbent les incidents réellement ponctuels. Elles ne doivent pas servir à masquer un scénario instable. Si un scénario n'atteint l'état `OK` qu'après une nouvelle tentative de façon régulière, analysez la cause.

Suivez dans le temps, pour chaque scénario, le taux d'exécutions en échec et le nombre de nouvelles tentatives, à partir de l'historique de la sonde ou de la télémétrie exportée (chapitres 17 et 18). Un scénario sain présente un taux d'échec nul en dehors des incidents réels.

Règles

- Traiter chaque échec intermittent comme un défaut à corriger.
- Limiter les nouvelles tentatives à une ou deux.
- Suivre le taux d'échec de chaque scénario.

16.15 Maintenir les scénarios au rythme des évolutions applicatives

Les applications évoluent : nouveaux libellés, formulaires réorganisés, étapes ajoutées au tunnel. Les scénarios doivent suivre. La génération précédente le rappelait déjà : les scripts doivent être maintenus à jour à chaque évolution de l'application web.

Organisation recommandée :

1. **Désigner un responsable** pour chaque scénario, côté exploitation ou côté équipe applicative.
2. **Intégrer les scénarios au cycle de mise en production.** Toute évolution touchant un parcours supervisé prévoit la mise à jour du scénario correspondant.
3. **Tester le scénario sur l'environnement de recette** avant la mise en production, avec une sonde ou le lanceur local.
4. **Pendant une intervention planifiée**, désactiver les scénarios concernés (chapitre 12, § 12.8) plutôt que de laisser des alertes être ignorées, et penser à les réactiver ensuite.
5. **Après la mise en production**, lancer une exécution d'essai, puis réactiver.
6. **Versionner** les scénarios scriptés dans un gestionnaire de code, et déployer par références immuables (chapitre 14, § 14.17).

> ✓ **Conseil** — Un échec `element_not_found` ou `assertion_failed` qui apparaît juste après une mise en production signale le plus souvent une évolution de la page, et non une panne. Comparez la capture d'écran au comportement attendu avant de déclarer un incident (chapitre 17).

Règles

- Un responsable par scénario.
- Mise à jour des scénarios intégrée au cycle de mise en production.
- Suspension planifiée pendant les interventions.

16.16 Collaborer avec les équipes de développement

Les scénarios les plus fiables sont conçus **avec** les équipes qui développent l'application. Quelques aménagements, peu coûteux pour elles, améliorent considérablement la supervision :

Demande	Bénéfice
Ajouter des attributs `data-testid` stables aux éléments des parcours critiques	Sélecteurs robustes, beaucoup moins de faux positifs
Exposer des valeurs techniques (`data-value="89.50"`) à côté des valeurs formatées	Assertions de calcul fiables
Fournir un compte de test, des produits de test et un moyen de paiement simulé	Parcours transactionnels sans risque
Exposer de façon fiable l'environnement (balise `meta` , en-tête)	Garde-fous des scénarios transactionnels
Purger côté serveur les données de test	Pas d'accumulation de données parasites
Prévenir des évolutions touchant les parcours supervisés	Scénarios mis à jour avant la mise en production
Exclure le trafic de la sonde des statistiques	Indicateurs métier non faussés
Instrumenter l'application avec OpenTelemetry	Corrélation possible entre les étapes du scénario et les traces côté serveur (chapitre 18)

Ces attributs et aménagements bénéficient aussi aux tests automatisés des équipes de développement. Ils constituent généralement une demande bien accueillie.

Règles

- Associer les équipes de développement à la conception des scénarios critiques.
- Formaliser les données de test et les attributs `data-testid` dans un document partagé.

16.17 Liste de contrôle avant la mise en service d'un scénario

#	Vérification	Référence
1	L'intention du scénario est documentée (ce qu'il prouve, résultat attendu, responsable)	§ 16.15
2	Le mode retenu est le plus simple possible (déclaratif si suffisant)	Chapitre 14, § 14.1.1
3	Les sélecteurs sont stables (`data-testid` de préférence), uniques, sans position ni XPath absolu	§ 16.1
4	Chaque étape importante se termine par une assertion portant sur le résultat métier	§ 16.2
5	Les messages d'assertion indiquent la valeur attendue et la valeur obtenue	§ 16.2
6	Aucune pause fixe ; les attentes de navigation et de réponse sont armées avant l'action	§ 16.4
7	Aucun `try/catch` qui masque une erreur	§ 16.5
8	Les noms d'étapes sont littéraux, stables et explicites	Chapitre 14, § 14.5.4
9	Aucune valeur sensible dans le code, les paramètres ou les journaux ; secrets déclarés au plus juste	§ 16.10
10	Le compte de test est dédié, limité en droits et connu des équipes	§ 16.7

#	Vérification	Référence
11	Les données créées sont annulées ou purgées ; aucune action irréversible hors environnement de test	§ 16.9
12	Les domaines tiers non essentiels sont bloqués	§ 16.11
13	Le trafic de la sonde est identifiable et exclu des statistiques	§ 16.12
14	Le scénario a été testé en succès **et** en échec sur poste de travail	Chapitre 14, § 14.15
15	Une exécution d'essai sur la sonde est conforme ; la durée mesurée sur la sonde est connue	Chapitre 14, § 14.16
16	Timeout, seuil d'avertissement et nouvelles tentatives sont fixés à partir de cette durée	§ 16.3
17	La fréquence est compatible avec la capacité de la sonde (un scénario à la fois par défaut)	Chapitre 13, § 13.8.3
18	Les équipes destinataires des alertes savent comment réagir	Chapitre 17

PARTIE V

Exploitation et maintenance

Chapitre 17 — Consultation et interprétation des résultats

Chapitre 18 — Intégration aux plateformes d'observabilité

Chapitre 19 — Diagnostic et résolution des problèmes

Chapitre 20 — Maintenance et mises à jour

Chapitre 17 — Consultation et interprétation des résultats

Ce chapitre explique où trouver les résultats produits par la sonde et comment les lire. Il propose ensuite une méthode pour transformer un résultat en diagnostic. Il s'adresse aux exploitants qui consultent la sonde au quotidien, et aux administrateurs qui automatisent la collecte des résultats.

Les notions de contrôle, d'exécution, d'état et de seuil sont définies au chapitre 8. Ce chapitre les suppose connues. La résolution des incidents propres à la sonde (démarrage, réseau, Chromium) est traitée au chapitre 19. La transmission des résultats vers une plateforme d'observabilité externe fait l'objet du chapitre 18.

> **Information** — Tous les noms d'hôtes, adresses et comptes de ce chapitre sont fictifs. La sonde d'exemple s'appelle `sonde-paris-01.example.internal`.

17.1 Où se trouvent les résultats

La sonde conserve elle-même l'intégralité de ses résultats. Elle ne dépend d'aucun serveur externe pour les produire ni pour les restituer : c'est la conséquence directe de son fonctionnement autonome (chapitre 3). Trois moyens de consultation coexistent.

Moyen	Public	Usage typique	Section
Interface d'administration	Exploitants, administrateurs	Consultation visuelle, analyse d'un échec, déclenchement manuel	§ 17.2
API REST	Administrateurs, outils d'automatisation	Intégration à des scripts, extraction de données, supervision par un outil tiers	§ 17.3
Plateformes d'observabilité	Équipes d'exploitation, centres de supervision	Vue multi-sondes, alertes, historique long terme	Chapitre 18

Ces trois moyens reposent sur les mêmes données, enregistrées dans la base locale de la sonde. Une exécution consultée dans l'interface, lue par l'API ou retrouvée sous forme de trace dans une plateforme externe a le même identifiant. On peut donc passer de l'un à l'autre sans ambiguïté.

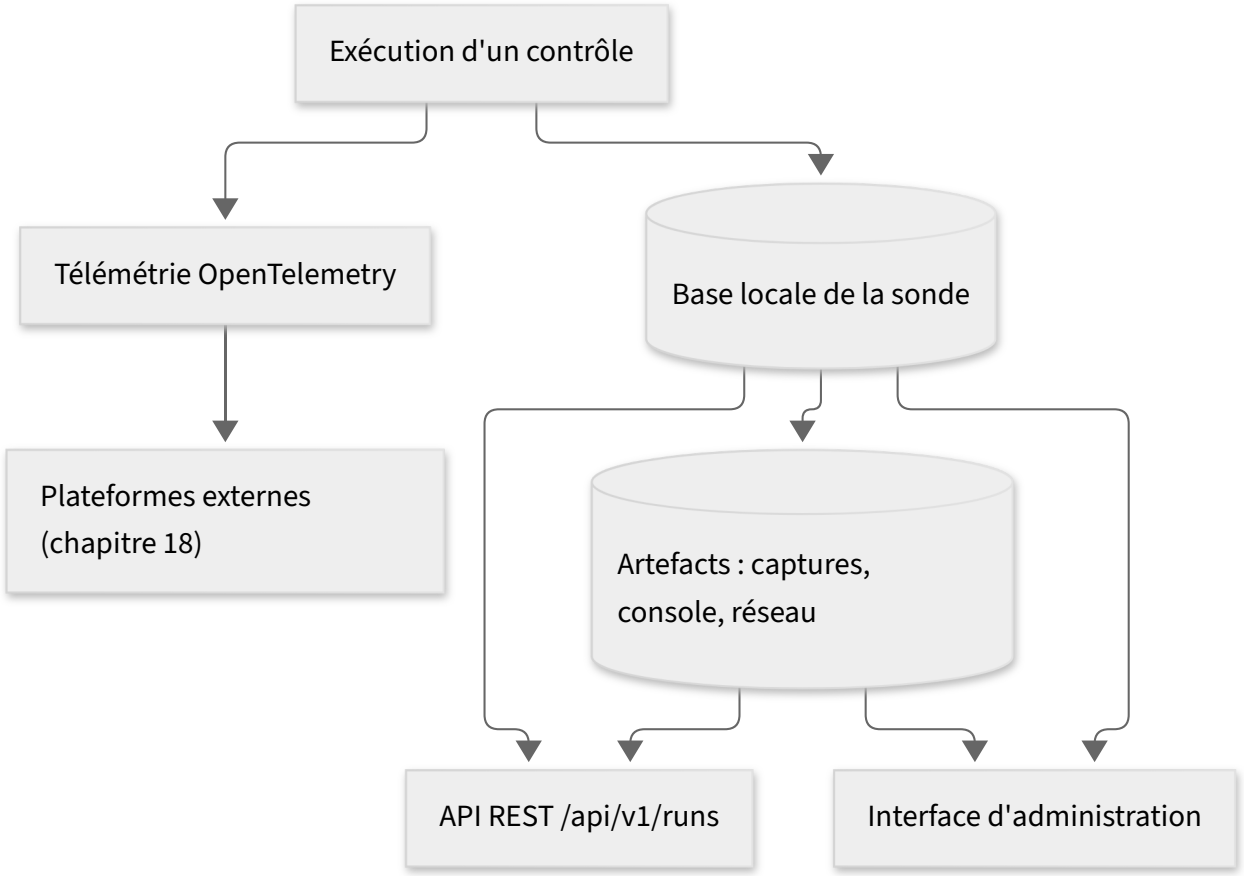


Schéma 17-1 — Où se trouvent les résultats

17.2 Consultation dans l'interface d'administration

L'interface d'administration est accessible depuis un navigateur à l'adresse `https://<adresse-ou-nom-de-la-sonde>/`, avec un compte disposant au minimum du rôle `viewer` (chapitre 9). Elle n'utilise que l'API publique de la sonde : tout ce qui y est affiché est également disponible par l'API (§ 17.3).

> **Information** — Les descriptions ci-dessous portent sur les informations présentées par l'interface d'administration, pas sur la disposition de ses écrans. Les mêmes informations sont accessibles par l'API REST.

17.2.1 Tableau de bord

Le tableau de bord est la page d'accueil de l'interface. Il répond en un coup d'œil à la question : « Y a-t-il un problème en ce moment ? »

Il présente :

- **une synthèse par état** : le nombre de contrôles actifs en `OK`, `WARNING`, `CRITICAL`, `UNKNOWN` et `STALE` ;
- **la liste des contrôles**, avec pour chacun son nom, son type, son état courant, la date du dernier changement d'état, la date de la dernière exécution et sa durée ;

- **des filtres** par état, par type de contrôle et par étiquette (`tags`, par exemple `application=boutique`) ;
- **l'état de santé de la sonde** elle-même (§ 17.9), signalé de manière distincte : un problème de la sonde n'est pas un problème des applications supervisées.

L'affichage se met à jour automatiquement à chaque fin d'exécution, grâce au flux d'événements décrit au § 17.3.6. Il n'est pas nécessaire de recharger la page.

> ✓ **Conseil** — Triez la liste par date de dernier changement d'état. Les contrôles qui viennent de passer en `CRITICAL` apparaissent ainsi en tête, avant ceux qui sont en échec depuis longtemps et déjà pris en charge.

17.2.2 Fiche d'un contrôle

En sélectionnant un contrôle, on accède à sa fiche. Elle regroupe :

- la définition du contrôle : cible, fréquence, timeout, nouvelles tentatives, seuils (lecture seule pour le rôle `viewer`) ;
- l'état courant et sa date de début ;
- l'historique des exécutions récentes, avec leur état et leur durée ;
- l'évolution de la durée d'exécution sur la période de rétention détaillée ;
- les actions disponibles selon le rôle : exécution immédiate, activation, désactivation (rôle `operator` au minimum).

17.2.3 Détail d'une exécution

Le détail d'une exécution est l'écran d'analyse par excellence. Il présente toutes les informations enregistrées pour cette exécution :

- l'état final, la durée totale, la date de début, le numéro de tentative et l'origine du lancement (planification, plan d'exécution, lancement manuel, nouvelle tentative, exécution d'essai) ;
- pour un échec : le **type d'erreur** (`error.type`), l'**étape** concernée et le **message** détaillé ;
- pour un scénario ou une suite de requêtes API : la **liste des étapes**, avec pour chacune son état et sa durée ;
- les **mesures** propres au type de contrôle : phases d'une requête HTTP, temps aller-retour ICMP, jours restants avant l'expiration d'un certificat, valeurs d'indicateurs métier, données de performance d'un plugin ;
- pour un plugin Nagios : la **sortie texte** du plugin ;
- les **artefacts** associés (§ 17.2.4) ;
- l'**identifiant de trace**, qui permet de retrouver la même exécution dans une plateforme d'observabilité (chapitre 18).

La liste des étapes se lit comme une frise chronologique. L'étape en échec est mise en évidence, et les étapes suivantes n'ont pas été exécutées : un scénario s'arrête à la première étape en échec. La durée de chaque étape permet de repérer une lenteur localisée, par exemple une authentification qui prend habituellement 2 secondes et qui en prend soudain 9.

17.2.4 Artefacts

Un artefact est un fichier produit par une exécution pour faciliter le diagnostic. Par défaut, les artefacts sont produits **uniquement en cas d'échec**, afin de limiter l'occupation du stockage et l'usure de la carte microSD.

Artefact	Type (`kind`)	Contenu	Contrôles concernés
Capture d'écran	`screenshot`	Image de la page au moment de l'échec	`browser`
Journal de console	`console`	Erreurs et avertissements JavaScript émis par la page	`browser`
Extrait réseau	`network`	Résumé des requêtes réseau de la page : adresses, statuts, durées, en nombre borné	`browser`
Code HTML	`html`	Contenu de la page au moment de l'échec, si activé	`browser`
Journal	`log`	Messages produits par le contrôle (`log.info`, `log.warn` dans un scénario scripté)	Tous

> ! **Avertissement** — Une capture d'écran ou un extrait de page peut contenir des données affichées par l'application supervisée. Le navigateur masque les champs de mot de passe, mais une page peut réafficher une information sensible (nom, adresse, numéro de commande). Utilisez exclusivement des comptes de test sans données réelles. Limitez l'accès aux artefacts aux personnes habilitées : rôle `viewer` au minimum, à attribuer avec discernement.

Les artefacts sont soumis à une double limite de rétention : 7 jours **et** 500 Mo par défaut (§ 17.8). La première limite atteinte déclenche la suppression des artefacts les plus anciens. Un artefact supprimé reste référencé dans le détail de l'exécution, mais n'est plus téléchargeable.

17.3 Consultation par l'API REST

L'API REST permet d'intégrer les résultats de la sonde à des outils tiers : scripts d'exploitation, outils de reporting, plateformes d'automatisation. Ses principes généraux (authentification, format des erreurs, pagination) sont présentés au chapitre 9. Sa référence complète figure à l'annexe F.

Les exemples ci-dessous utilisent `curl`, disponible sur la plupart des systèmes, et un jeton d'API de rôle `viewer` lu dans le fichier d'en-tête protégé `$HOME/.sva/entete-auth`, créé une fois selon le chapitre 9 (§ 9.3.3) : le jeton n'apparaît jamais dans les arguments d'une commande, ni dans l'historique du shell. Le fichier `sonde-ca.pem` contient le certificat de la sonde, récupéré et vérifié selon le chapitre 6 (§ 6.4.4), ou celui de l'autorité de certification qui l'a émis (chapitre 7).

```bash
# Variables utilisées dans les exemples de ce chapitre (valeurs fictives),
# définies comme au § 9.3.3 : adresse de la sonde et certificat vérifié
export SONDE="https://sonde-paris-01.example.internal"
export CA="$HOME/.sva/sonde-paris-01.pem"
# Le jeton n'est pas placé dans une variable : il est lu dans le fichier
# "$HOME/.sva/entete-auth" (droits 600), créé une fois selon le § 9.3.3.
```

> ! **Attention** — N'utilisez pas l'option `-k` (`--insecure`) de `curl` pour interroger l'API. Elle désactive la vérification du certificat et expose le jeton à une interception. Utilisez `--cacert` avec le certificat de la sonde ; la seule exception est le téléchargement initial de ce certificat, immédiatement suivi de la vérification de son empreinte (chapitre 6, § 6.4.4).

17.3.1 Dernier état d'un contrôle

La ressource `GET /api/v1/checks/{id}/status` renvoie l'état consolidé d'un contrôle. Elle est accessible avec un jeton de rôle `metrics`, le plus restreint, qui convient aux outils de supervision tiers comme le plugin Nagios `check_sva` (chapitre 18).

```bash
curl --cacert "$CA" \
     -H @"$HOME/.sva/entete-auth" \
     "$SONDE/api/v1/checks/boutique-connexion/status"
```

Exemple de réponse :

```json
{
  "checkId": "boutique-connexion",
  "status": "CRITICAL",
  "since": "2026-11-19T09:15:16.764Z",
  "lastRun": {
    "id": "01M7SHD1K2X7Q9N3V4T6W8Y0ZA",
    "startedAt": "2026-11-19T09:15:02.114Z",
    "durationMs": 14650,
    "errorType": "assertion_failed",
    "step": "Vérifier l'accès à l'espace client"
  }
}
```

> **Information** — Les champs de cette réponse (`checkId`, `status`, `since`, `lastRun`) sont décrits dans la description OpenAPI de la sonde (`GET /api/v1/openapi.json`).

17.3.2 Historique des exécutions

La ressource `GET /api/v1/runs` renvoie l'historique des exécutions, du plus récent au plus ancien. L'ordre est **total** : les exécutions sont triées par date de début (`startedAt`), puis par identifiant (`id`) en cas d'égalité. Elle accepte les filtres suivants.

Paramètre	Description	Exemple
`checkId`	Limite aux exécutions d'un contrôle	`checkId=boutique-connexion`
`status`	Limite à un état	`status=CRITICAL`
`from`, `to`	Bornes de la période (RFC 3339, UTC)	`from=2026-11-19T00:00:00Z`
`limit`	Nombre d'éléments par page	`limit=100`
`cursor`	Curseur de la page suivante, fourni par la réponse précédente	`cursor=eyJ…`

Exemple : les exécutions en échec d'un contrôle sur une journée.

```bash
curl --cacert "$CA" \
     -H @"$HOME/.sva/entete-auth" \
     "$SONDE/api/v1/runs?checkId=boutique-connexion&status=CRITICAL&from=2026-11-
    19T00:00:00Z&to=2026-11-20T00:00:00Z&limit=100"
```

La réponse a la forme suivante :

```json
{
  "items": [
    { "id": "01M7SHD1K2X7Q9N3V4T6W8Y0ZA", "checkId": "boutique-connexion", "status": "CRITICAL",
    "startedAt": "2026-11-19T09:15:02.114Z", "durationMs": 14650 }
  ],
  "nextCursor": "eyJiZWZvcmUiOiIwMUpEQjVLOFIy…"
}
```

Pagination. Lorsque la réponse contient `nextCursor`, d'autres résultats existent : répétez la requête avec `cursor=<valeur de nextCursor>`, sans modifier les autres paramètres, jusqu'à ce que `nextCursor` soit absent.

Garanties de lecture (à l'intention des intégrateurs). Les règles suivantes concernent les scripts d'export et les outils qui doivent obtenir un ensemble exhaustif ou reproductible. La pagination par curseur garantit qu'aucune exécution n'est répétée et qu'aucune exécution déjà enregistrée au début de la lecture n'est omise, sous réserve de la rétention. Une exécution enregistrée **pendant** la lecture peut en revanche manquer (ligne « Exécutions nouvelles »).

Règle	Comportement
Ordre	Ordre total (`startedAt`, `id`) : deux exécutions de même date de début restent départagées, et chaque page reprend exactement après la dernière exécution lue
Curseur	Il encode la dernière position lue. Il reste valide **24 heures** ; au-delà, la sonde répond `410 Gone` et la lecture doit reprendre depuis la première page
Exécutions nouvelles	L'historique est lu du plus récent au plus ancien, et la position d'une exécution dépend de sa date de **début** (`startedAt`), et non de sa date d'enregistrement. Une exécution enregistrée pendant la lecture n'est donc renvoyée que si sa date de début est antérieure à la dernière position lue. Sinon, qu'il s'agisse d'une exécution nouvelle ou d'une exécution longue qui s'achève pendant la lecture, elle n'apparaît que dans une lecture ultérieure
Borne de fin `to`	Pour un **export reproductible**, fixez toujours `to` : les exécutions commencées après cette borne sont exclues. Une exécution commencée **avant** `to` mais enregistrée plus tard reste en revanche dans la période et relève de la ligne précédente. Deux lectures successives de la même période ne donnent donc le même ensemble que si `to` précède l'instant de lecture d'au moins la durée maximale d'une exécution, de sorte qu'aucune exécution de la période ne reste en cours, et sous réserve de la rétention
Rétention	Les exécutions purgées par la rétention (§ 17.8) **pendant** la lecture disparaissent des pages suivantes
Instantané	La sonde ne fournit pas d'instantané : la lecture reflète l'état des données au moment de chaque page

> **Attention** — Pour une période proche de la limite de rétention (14 jours par défaut, § 17.8), les exécutions les plus anciennes peuvent être purgées entre deux pages. Pour un export exhaustif, choisissez une période nettement plus récente que cette limite, ou lisez-la en une seule session, sans interruption.

17.3.3 Détail d'une exécution

La ressource `GET /api/v1/runs/{runId}` renvoie le résultat complet d'une exécution, au format décrit au § 17.4.

```bash
curl --cacert "$CA" \
     -H @"$HOME/.sva/entete-auth" \
     "$SONDE/api/v1/runs/01M7SHD1K2X7Q9N3V4T6W8Y0ZA"
```

17.3.4 Téléchargement des artefacts

Chaque artefact listé dans le détail d'une exécution est téléchargeable par `GET /api/v1/runs/{runId}/artifacts/{name}`.

```bash
# Télécharger la capture d'écran de l'échec
curl --cacert "$CA" \
     -H @"$HOME/.sva/entete-auth" \
     -o failure.png \
     "$SONDE/api/v1/runs/01M7SHD1K2X7Q9N3V4T6W8Y0ZA/artifacts/failure.png"
```

Un artefact purgé par la rétention renvoie une erreur `404`, au format `application/problem+json`.

17.3.5 Exécution manuelle pour confirmer un incident

Avant d'escalader un incident, il est souvent utile de relancer immédiatement le contrôle pour vérifier que l'échec persiste. Cette action requiert le rôle `operator` : le fichier d'en-tête utilisé doit contenir un jeton de ce rôle. L'exemple suppose un second fichier, `$HOME/.sva/entete-auth-operator`, créé selon la même procédure (§ 9.3.3) avec un jeton de rôle `operator`.

```bash
curl --cacert "$CA" -X POST \
     -H @"$HOME/.sva/entete-auth-operator" \
     "$SONDE/api/v1/checks/boutique-connexion/runs"
```

La sonde répond `202 Accepted`, avec un en-tête `Location` qui désigne l'exécution créée. Interrogez ensuite cette adresse pour obtenir le résultat de l'exécution. Une exécution manuelle est enregistrée et exportée comme toute autre exécution. Son origine (`trigger`) vaut `manual`.

Pour tester un contrôle sans modifier son état ni alimenter les plateformes externes, utilisez plutôt l'exécution d'essai `?dryRun=true`, décrite au chapitre 9 (§ 9.6.5).

17.3.6 Suivi en temps réel

La ressource `GET /api/v1/events` diffuse les événements de la sonde au format **Server-Sent Events** (SSE). Avec ce mécanisme standard, le client garde une connexion HTTPS ouverte et reçoit les événements au fil de l'eau, sans interroger la sonde à intervalles réguliers.

Événement	Émis lorsque
`run.started`	Une exécution démarre
`run.finished`	Une exécution se termine (avec son état)
`check.status.changed`	L'état consolidé d'un contrôle change, par exemple de `OK` à `CRITICAL`
`system.alert`	La sonde signale un problème la concernant : stockage, mémoire, température, export

```bash
# Suivre les événements en direct (Ctrl+C pour arrêter)
curl --cacert "$CA" -N \
     -H @"$HOME/.sva/entete-auth" \
     "$SONDE/api/v1/events"
```

Le flux d'événements sert au suivi interactif et à l'automatisation légère. Pour l'alerte de production, privilégiez les mécanismes du chapitre 18 : ils sont conçus pour la haute disponibilité et la gestion des notifications.

17.4 Lire un résultat d'exécution

Le résultat d'une exécution suit un format unique, quel que soit le type de contrôle. L'exemple ci-dessous est celui du scénario déclaratif `boutique-connexion` (chapitre 14, § 14.3.1) en échec, sur une boutique en ligne fictive ; c'est le même que celui du chapitre 8 (§ 8.7.2).

```json
{
  "id": "01M7SHD1K2X7Q9N3V4T6W8Y0ZA",
  "checkId": "boutique-connexion",
  "checkRevision": 7,
  "planRunId": null,
  "trigger": "retry",
  "status": "CRITICAL",
  "attempt": 2,
  "startedAt": "2026-11-19T09:15:02.114Z",
  "durationMs": 14650,
  "error": {
    "type": "assertion_failed",
    "step": "Vérifier l'accès à l'espace client",
    "message": "Élément [data-testid=account-menu] non visible après 10000 ms"
  },
  "steps": [
    { "index": 0, "name": "Ouvrir la page de connexion", "status": "OK", "startedOffsetMs": 50,
    "durationMs": 1830 },
    { "index": 1, "name": "Saisir l'identifiant", "status": "OK", "startedOffsetMs": 1880,
    "durationMs": 160 },
    { "index": 2, "name": "Saisir le mot de passe", "status": "OK", "startedOffsetMs": 2040,
    "durationMs": 140 },
    { "index": 3, "name": "Valider le formulaire", "status": "OK", "startedOffsetMs": 2180,
    "durationMs": 2410 },
    { "index": 4, "name": "Vérifier l'accès à l'espace client", "status": "CRITICAL",
    "startedOffsetMs": 4590, "durationMs": 10004 }
  ],
  "measurements": { "browser.launch_ms": 1180 },
  "artifacts": [
    { "name": "failure.png", "kind": "screenshot", "bytes": 182344 },
    { "name": "console.json", "kind": "console", "bytes": 2210 }
  ],
  "trace": { "traceId": "4bf92f3577b34da6a3ce929d0e0e4736" }
}
```

Champ	Signification	Utilité pour l'analyse
`id`	Identifiant unique de l'exécution (format ULID, triable chronologiquement)	Référence à citer dans un ticket d'incident
`checkId`	Identifiant du contrôle	—
`checkRevision`	Version de la configuration du contrôle utilisée	Vérifier si une modification récente du contrôle coïncide avec l'apparition de l'échec
`planRunId`	Exécution du plan d'exécution dont fait partie celle-ci, ou `null`	Relier les contrôles enchaînés d'un même parcours (chapitre 12)
`trigger`	Origine : `schedule`, `plan`, `manual`, `retry`, `dryRun`	Distinguer une exécution planifiée d'un lancement manuel ; ici `retry`, nouvelle tentative après un premier échec
`status`	État final de l'exécution	—
`attempt`	Numéro de tentative (1 = première)	Ici 2 : la première tentative avait déjà échoué ; l'échec est confirmé (§ 17.6)
`startedAt`	Date et heure de début, en UTC	Corrélation avec les journaux de l'application

Champ	Signification	Utilité pour l'analyse
`durationMs`	Durée totale en millisecondes	—
`error.type`	Catégorie d'erreur, parmi une liste fermée (§ 17.5.2)	Classer et agréger les échecs
`error.step`	Étape en échec	Localiser le problème dans le parcours
`error.message`	Message détaillé, secrets masqués	Comprendre la cause précise
`steps[]`	Étapes avec état, décalage de début et durée	Repérer une lenteur localisée ; ici, l'étape en échec dure un peu plus de 10 secondes, le délai d'attente de l'élément
`measurements`	Mesures propres au type de contrôle	Diagnostic technique (§ 17.5.4)
`output`	Sortie texte d'un plugin Nagios (absent ici)	—
`artifacts[]`	Artefacts disponibles	Capture d'écran, console
`trace.traceId`	Identifiant de trace OpenTelemetry	Retrouver l'exécution dans une plateforme externe

> **Information** — Toutes les dates sont exprimées en UTC dans l'API. L'interface d'administration les affiche dans le fuseau horaire configuré sur la sonde. Pensez-y lorsque vous comparez avec les journaux d'une application, qui sont souvent en heure locale.

Pour les autres types de contrôles, les mesures de `measurements` sont décrites dans les chapitres correspondants : chapitre 10 pour les contrôles réseau et techniques, chapitre 11 pour les contrôles applicatifs et métier. Quelques exemples courants :

Type	Exemples de mesures
`http`	Code de statut, durée des phases DNS, connexion, TLS, premier octet et transfert, jours restants avant l'expiration du certificat
`icmp`	Temps aller-retour moyen, taux de perte
`dns`	Durée de résolution, réponses obtenues
`database`	Valeurs des indicateurs exportés, par exemple le nombre de commandes du jour
`nagios`	Données de performance déclarées exportables
`browser`	Durée de lancement du navigateur (`browser.launch_ms`), exclue de la durée de l'exécution (§ 17.7.1)

17.5 Méthode d'analyse d'un échec

Face à un contrôle en échec, la tentation est d'ouvrir immédiatement la capture d'écran. Une démarche structurée en cinq questions permet d'aller plus vite et d'éviter les conclusions hâtives.

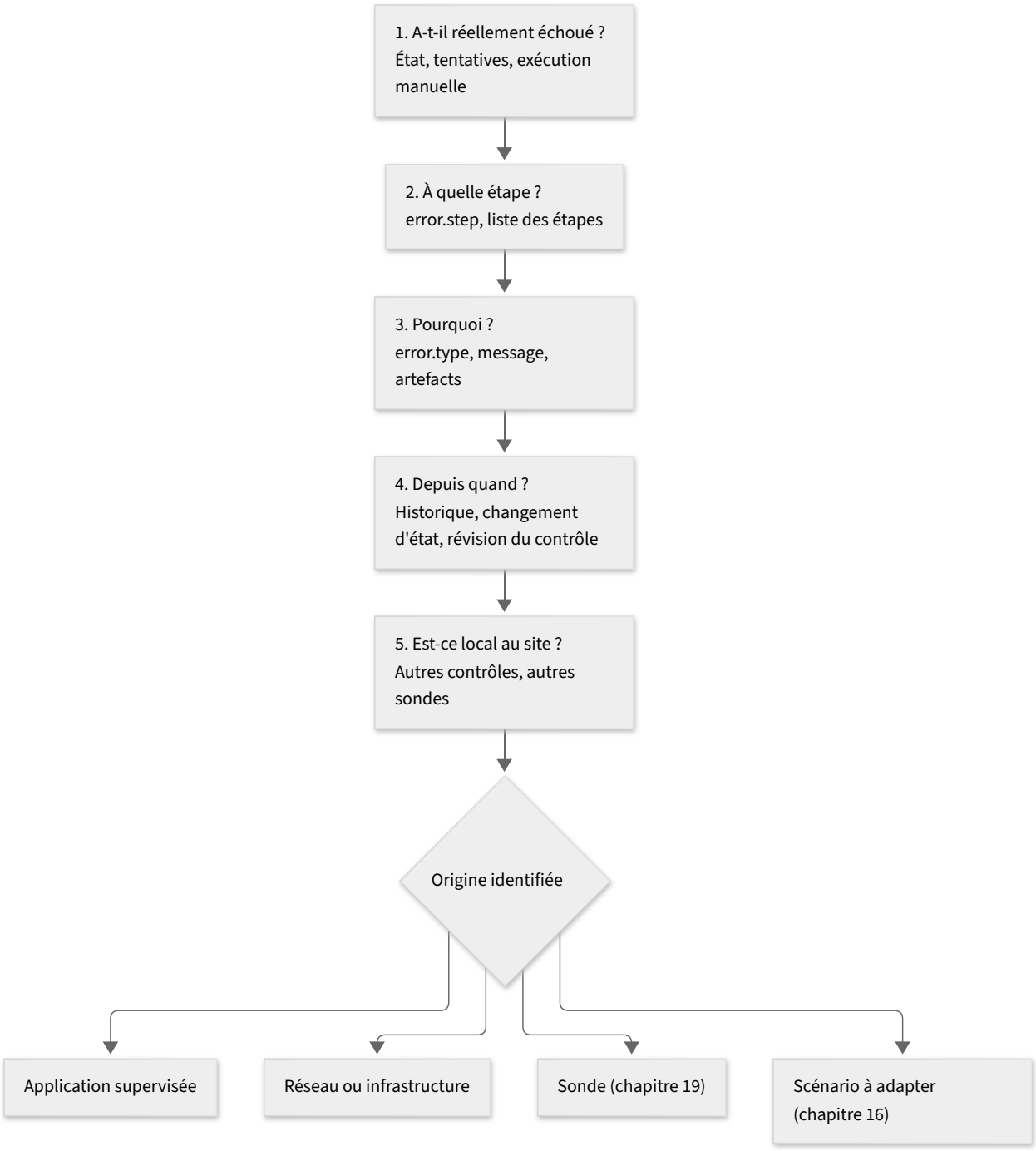


Schéma 17-2 — Méthode d'analyse d'un échec en cinq questions

17.5.1 A-t-il réellement échoué ?

Commencez par qualifier l'état.

- `CRITICAL` : la cible n'a pas satisfait les critères du contrôle. C'est un signal sur l'application ou l'infrastructure supervisée.

- `UNKNOWN` : le contrôle **n'a pas pu être évalué**. La cause se trouve en général du côté de la sonde ou de la configuration du contrôle (navigateur qui ne démarre pas, plugin introuvable, ressources insuffisantes), et non du côté de l'application. Un `UNKNOWN` de type `maintenance` survenu dans une fenêtre de maintenance des mises à jour est normal : l'exécution a été interrompue par le drainage qui précède l'activation (chapitre 8, § 8.14.2). Reportez-vous au chapitre 19.
- `WARNING` : le contrôle a réussi, mais un seuil de performance est dépassé, ou un plugin Nagios a retourné le code 1. L'application fonctionne, mais lentement.
- `STALE` : aucun résultat n'a été produit depuis deux intervalles. Le contrôle ne s'exécute plus, ce qui oriente vers la sonde (§ 17.9 et chapitre 19).

Vérifiez ensuite le champ `attempt`. Si le contrôle prévoit des nouvelles tentatives et que l'état `CRITICAL` survient à la dernière tentative, l'échec a été confirmé plusieurs fois (§ 17.6). En cas de doute, lancez une exécution manuelle (§ 17.3.5).

17.5.2 À quelle étape et pourquoi ?

Le champ `error.step` désigne l'étape en échec, et `error.type` la catégorie de l'erreur, choisie dans une liste fermée. Le tableau suivant est la **référence** de ces catégories : il donne, pour chacune, l'état produit, l'origine la plus probable et la piste à suivre en priorité. Les règles de calcul de l'état sont décrites au chapitre 8 (§ 8.8.2).

`error.type`	Signification	État	Origine la plus probable	Première vérification
`timeout`	Le délai maximal de l'exécution ou d'une attente a été dépassé	`CRITICAL`	Application lente ou bloquée, flux filtré sans rejet ; parfois timeout trop court	Durée habituelle de l'étape dans l'historique
`dns_error`	Le nom de la cible ne se résout pas	`CRITICAL`	Serveurs DNS de la sonde, nom erroné, enregistrement supprimé	Résolution depuis un autre poste du même réseau
`connection_refused`	La cible refuse la connexion	`CRITICAL`	Service arrêté, mauvais port, pare-feu rejetant activement	État du service sur le serveur
`connection_reset`	Connexion interrompue par la cible ou un équipement intermédiaire	`CRITICAL`	Pare-feu, répartiteur de charge, saturation	Journaux des équipements réseau
`tls_error`	Échec de la négociation TLS ou certificat non reconnu	`CRITICAL`	Autorité interne non importée, nom ne correspondant pas au certificat, protocole obsolète	Chaîne de certification (chapitre 7)
`certificate_expired`	Certificat de la cible expiré	`CRITICAL`	Renouvellement oublié	Date d'expiration du certificat
`http_status`	Code HTTP différent de celui attendu	`CRITICAL`	Erreur applicative (5xx), ressource déplacée (404), accès refusé (401, 403)	Code obtenu dans le message
`assertion_failed`	Une assertion ou une règle n'est pas satisfaite	`CRITICAL` (`WARNING` pour une règle de sévérité `warning`)	Dysfonctionnement fonctionnel, valeur métier hors seuil, ou évolution de l'application	Capture d'écran, contenu attendu, message

`error.type`	Signification	État	Origine la plus probable	Première vérification
`element_not_found`	Un élément attendu de la page est introuvable ou invisible	`CRITICAL`	Page modifiée (sélecteur à mettre à jour), page d'erreur affichée, lenteur	Capture d'écran et code HTML
`navigation_error`	La navigation vers une page a échoué	`CRITICAL`	Site injoignable depuis le navigateur, redirection en boucle	Extrait réseau
`script_error`	Erreur de programmation dans un scénario scripté : le parcours n'a pas pu être vérifié	`CRITICAL`	Script à corriger	Message et numéro de ligne ; exécution d'essai après correction (chapitre 9, § 9.6.5)
`database_error`	Erreur lors de la connexion ou de la requête SQL	`CRITICAL`	Base indisponible, identifiants, requête invalide, droits du compte	Message du pilote
`auth_failed`	Authentification refusée par la cible	`CRITICAL`	Compte de test verrouillé, mot de passe expiré ou modifié	Validité du secret (chapitre 9)
`browser_crash`	Le navigateur Chromium s'est arrêté anormalement	`UNKNOWN`	Mémoire insuffisante sur la sonde, page très lourde	Santé de la sonde (§ 17.9), chapitre 19
`plugin_error`	Le plugin Nagios n'a pas pu s'exécuter jusqu'à son terme, ou a renvoyé un code retour en dehors de 0 à 3	`UNKNOWN`	Plugin absent ou non exécutable, arguments invalides, plugin interrompu	Sortie du plugin, nom et arguments (chapitre 10)
`no_data`	Un contrôle `database` n'a renvoyé aucune ligne : les règles ne peuvent pas être évaluées	`UNKNOWN` par défaut, selon `onEmpty`	Requête sans agrégation, filtre trop restrictif, données absentes sur la période	Requête du contrôle (chapitre 11)
`secret_missing`	Un secret référencé par le contrôle est absent de la sonde	`UNKNOWN`	Secret non créé, supprimé ou mal orthographié dans le contrôle ; configuration importée sans ses secrets	Liste des secrets de la sonde et noms déclarés dans `secrets` (chapitre 9, § 9.8)
`resource_exhausted`	Ressources de la sonde insuffisantes (mémoire, processus)	`UNKNOWN`	Trop d'exécutions simultanées, scénario trop gourmand	Santé de la sonde
`runner_crash`	Le moteur de scénarios s'est arrêté pendant l'exécution	`UNKNOWN`	Incident de la sonde	Chapitre 19
`agent_restart`	Le moteur principal ou la sonde a redémarré pendant l'exécution	`UNKNOWN`	Redémarrage, coupure électrique, mise à jour ; normal s'il est ponctuel	Journal d'audit, historique de mise à jour

`error.type`	Signification	État	Origine la plus probable	Première vérification
`maintenance`	L'exécution a été interrompue au début de l'activation d'une mise à jour, faute de s'être terminée dans le délai de drainage (`updates.drainTimeout`)	`UNKNOWN`	Activation d'une mise à jour dans une fenêtre de maintenance, ou à la demande d'un administrateur	Date de l'exécution comparée aux fenêtres (`updates.windows`), journal d'audit, événements `maintenance.started` et `maintenance.ended` ; s'il se répète, allonger `drainTimeout` (30 minutes au maximum) ou déplacer la fenêtre hors des plans longs (chapitre 20)
`unknown`	Erreur non classée	`UNKNOWN`	—	Message détaillé et journaux ; support si l'erreur se répète

Le message `error.message` complète la catégorie avec les détails : élément attendu, code HTTP reçu, délai écoulé. Pour les scénarios navigateur, la **capture d'écran** confirme souvent le diagnostic en quelques secondes. Elle révèle par exemple une page de maintenance, une bannière de consentement qui masque le bouton attendu, ou un message d'erreur de l'application.

> ✓ **Conseil** — Un échec `element_not_found` ou `assertion_failed` qui apparaît juste après une mise en production de l'application supervisée signale souvent une **évolution de la page**, et non une panne. Comparez la capture d'écran avec le comportement attendu avant de déclarer un incident. Le chapitre 16 explique comment écrire des scénarios robustes à ces évolutions.

17.5.3 Depuis quand ?

La date de début de l'état courant (`since` dans `/status`, date du dernier changement d'état dans l'interface) indique depuis quand le problème est observé. Trois vérifications complètent cette information :

1. **L'historique des exécutions** : l'échec est-il continu, ou intermittent (§ 17.6) ?
2. **La révision du contrôle** (`checkRevision`) : si elle a changé au moment où l'échec est apparu, la modification du contrôle est la première suspecte. Le journal d'audit (chapitre 9) indique qui l'a modifié et quand.
3. **Les événements extérieurs** : une mise en production, une maintenance réseau ou un changement de mot de passe du compte de test à la même heure.

17.5.4 Est-ce local au site ?

Une sonde observe les applications **depuis l'emplacement où elle est installée**. Un échec peut donc refléter un problème de l'application, ou un problème du chemin réseau entre la sonde et l'application.

Observation	Interprétation probable
Tous les contrôles d'une même application échouent, les autres sont `OK`	Problème de l'application
Tous les contrôles de la sonde échouent ou passent en `UNKNOWN`	Problème de la sonde ou de son raccordement réseau (chapitre 19)
Seuls les contrôles vers des destinations externes (Internet) échouent	Problème de sortie Internet du site : proxy, pare-feu, lien opérateur
Une autre sonde, installée sur un autre site, voit la même application `OK`	Problème du chemin réseau propre au site

Observation	Interprétation probable
Les mesures HTTP montrent une phase `dns` ou `connect` anormalement longue	Réseau ou résolution de noms, plutôt que l'application
La phase `ttfb` (attente du premier octet) est longue	Temps de traitement côté serveur

La comparaison entre plusieurs sondes installées sur des sites différents est le moyen le plus sûr de distinguer une panne applicative d'un problème local. Elle s'effectue dans une plateforme d'observabilité grâce à l'attribut `sva.site` (chapitre 18).

17.6 Bagotage, nouvelles tentatives et confirmation des incidents

On parle de **bagotage** lorsqu'un contrôle alterne rapidement entre les états `OK` et `CRITICAL`. Le bagotage a plusieurs origines : instabilité réelle de l'application, saturation intermittente du réseau, ou scénario fragile qui échoue de manière aléatoire. Il génère des alertes à répétition et finit par décrédibiliser la supervision.

La sonde dispose d'un mécanisme de **nouvelles tentatives** (`retry`, chapitre 8). Lorsqu'une exécution échoue, la sonde relance le contrôle jusqu'à `retry.attempts` fois, à `retry.interval` d'intervalle, avant de retenir l'état `CRITICAL`. Le champ `attempt` du résultat indique la tentative concernée.

Configuration	Effet	Usage conseillé
`attempts: 0`	Le premier échec déclenche l'état `CRITICAL`	Contrôles très fiables, ou lorsque la moindre indisponibilité doit être signalée
`attempts: 1`, `interval: 30s`	L'échec doit se répéter 30 secondes plus tard	Valeur raisonnable pour la plupart des scénarios navigateur
`attempts: 2` ou plus	Réduit fortement les alertes sur échecs isolés	Applications connues pour des micro-coupures ; au prix d'un délai de détection plus long

> **Attention** — Les nouvelles tentatives allongent le délai de détection d'un incident réel : avec un intervalle de 5 minutes, 2 tentatives espacées de 30 secondes, il faut jusqu'à 6 minutes pour que l'état passe à `CRITICAL`. Recherchez l'équilibre entre réactivité et fiabilité des alertes, et ne masquez pas un scénario fragile par des tentatives multiples : corrigez-le (chapitre 16).

Pour identifier le bagotage, consultez l'historique d'un contrôle sur 24 heures. Un nombre élevé de changements d'état, ou d'exécutions réussies à la deuxième tentative, est un signal à examiner. Dans une plateforme d'observabilité, le compteur `sva.check.runs` (chapitre 18) n'en donne qu'un indice indirect, dans les limites décrites ci-dessous ; la mesure elle-même repose sur l'historique des exécutions et sur les événements de changement d'état.

Ce compteur mesure exactement le **nombre d'exécutions exportées**, ventilé par contrôle (`sva.check.id`), type de contrôle (`sva.check.type`), état de l'exécution (`sva.check.status`) et type d'erreur (`error.type`). Une hausse de la part des exécutions `CRITICAL` alors que l'état consolidé reste `OK` ne révèle des échecs non confirmés que si les tentatives non confirmées sont comptées dans ce compteur ; les notes de version précisent ce point. En outre, il ne porte ni numéro de tentative, ni état précédent, ni identifiant d'incident : il ne permet pas, à lui seul, de compter les réussites obtenues à la deuxième tentative ni le nombre de changements d'état. Pour ces deux mesures, utilisez :

- l'**historique des exécutions** (`GET /api/v1/runs`, § 17.3.2), dont chaque élément détaillé porte `attempt` et `trigger` (§ 17.4) : une exécution `OK` de tentative 2 ou plus, précédée d'un échec, est une réussite en nouvelle tentative ;
- les **événements** `check.status.changed` (§ 17.3.6), émis à chaque changement de l'état consolidé : leur nombre sur une période est le nombre de transitions d'état.

17.7 Indicateurs : temps de réponse, percentiles et disponibilité

17.7.1 Temps de réponse

La durée d'exécution d'un contrôle mesure le temps nécessaire pour accomplir le parcours ou la requête **depuis la sonde**. Elle inclut le temps réseau entre la sonde et l'application. Elle peut donc différer des temps mesurés côté serveur ou par un utilisateur situé ailleurs.

Pour un scénario navigateur, la durée totale inclut le chargement des pages, l'exécution du JavaScript de l'application et les attentes explicites du scénario. Elle exclut le lancement du navigateur, mesuré séparément (`browser.launch_ms` dans `measurements`) : une lenteur de Chromium sur la sonde n'est ainsi pas imputée à l'application. La durée de chaque étape est plus parlante que la durée totale : elle isole la partie du parcours qui se dégrade.

17.7.2 Moyenne et percentiles

La **moyenne** des temps de réponse est sensible aux valeurs extrêmes et masque les dégradations ponctuelles. Les **percentiles** décrivent mieux l'expérience réelle.

- Le **50e percentile** (p50, ou médiane) est la durée en dessous de laquelle se trouvent la moitié des exécutions. Il représente le cas typique.
- Le **95e percentile** (p95) est la durée en dessous de laquelle se trouvent 95 % des exécutions. Il représente les cas les plus lents, hors valeurs exceptionnelles.

Exemple : sur 100 exécutions d'un parcours, 94 durent environ 4 secondes et 6 durent environ 20 secondes. La moyenne vaut environ 5 secondes, ce qui semble acceptable. Le p95, lui, est proche de 20 secondes, ce qui révèle que 6 % des parcours sont très lents.

La sonde calcule les percentiles p50 et p95 dans ses **agrégats horaires** (§ 17.8). Les plateformes d'observabilité les calculent à partir des histogrammes de durée exportés (chapitre 18).

17.7.3 Disponibilité calculée

La disponibilité d'un contrôle sur une période est la proportion d'exécutions réussies parmi les exécutions **évaluables**. Il s'agit d'une proportion d'**exécutions**, et non d'une proportion du **temps** : les périodes sans mesure (sonde arrêtée, maintenance, échéances sautées) et les changements de fréquence peuvent dissocier les deux. La formule recommandée est la suivante :

```
TEXTE
                          nombre d'exécutions OK + WARNING
Disponibilité = ──────────────────────────────────────────────────
                  nombre d'exécutions OK + WARNING + CRITICAL
```

Deux choix sont implicites dans cette formule :

- **`WARNING` compte comme disponible** : l'application a fonctionné, même lentement. Pour un engagement de performance, calculez un indicateur distinct sur les durées.
- **`UNKNOWN` est exclu du calcul** : une exécution `UNKNOWN` signifie que la sonde n'a pas pu évaluer l'application. La compter comme une indisponibilité imputerait à l'application un problème de la sonde, et la compter comme disponible masquerait une absence de mesure. Le nombre d'exécutions `UNKNOWN` doit être suivi séparément, comme indicateur de **qualité de mesure**.

Exemple chiffré. Un contrôle s'exécute toutes les 5 minutes pendant 30 jours, soit 8 640 exécutions planifiées. On observe 8 590 exécutions `OK`, 12 `WARNING`, 26 `CRITICAL` et 12 `UNKNOWN`.

```texte
Exécutions évaluables  = 8 590 + 12 + 26          = 8 628
Disponibilité          = (8 590 + 12) / 8 628      = 99,70 %
Qualité de mesure      = 8 628 / 8 640             = 99,86 %
```

La durée d'indisponibilité correspondante peut s'estimer en multipliant le nombre d'exécutions `CRITICAL` par l'intervalle : 26 × 5 minutes, soit environ 2 heures et 10 minutes. Cette estimation est d'autant plus précise que l'intervalle est court.

Publiez toujours la disponibilité avec sa **couverture** : la part des échéances attendues qui ont donné une exécution évaluable (ici la qualité de mesure, 99,86 %). Une disponibilité de 99,70 % signifie « 99,70 % des exécutions évaluables ont réussi », et non « l'application a été disponible 99,70 % du temps ». Une disponibilité **temporelle** ne peut être déduite de ces chiffres que par une méthode explicitement convenue avec les parties prenantes (par exemple, attribution à chaque exécution de l'intervalle qui la suit, et traitement des périodes non couvertes).

Calcul dans Prometheus. Les requêtes suivantes calculent la même formule à partir du compteur `sva_check_runs_total` (chapitre 18, § 18.3.3). Elles respectent quatre règles :

- **Vue par sonde** : grouper par `job`, `instance` et `sva_check_id`. Deux sondes peuvent porter un contrôle de même identifiant ; grouper par `sva_check_id` seul mélangerait leurs mesures.
- **Vue de parc** : une agrégation distincte, explicitement nommée, additionne les exécutions de toutes les sondes pour un même identifiant de contrôle. Elle n'a de sens que si ces contrôles mesurent la même cible.
- **États valides** : les états évaluables sont sélectionnés explicitement (`sva_check_status=~"OK|WARNING|CRITICAL"`). Un filtre d'exclusion tel que `sva_check_status!="UNKNOWN"` sélectionnerait aussi une série dépourvue de label d'état.
- **Absence de numérateur et dénominateur nul** : un contrôle sans aucune exécution réussie n'a pas de série `OK` ; sans précaution, la division ne renvoie rien au lieu de 0 %. Le terme `or 0 * (…)` complète le numérateur par zéro lorsque des exécutions évaluables existent. À l'inverse, sans aucune exécution évaluable (dénominateur nul), la condition `> 0` fait disparaître le résultat : « aucune mesure » n'est ni 0 % ni 100 %.

```promql
# Disponibilité sur 30 jours, par sonde et par contrôle
(
    sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING"}[30d]))
  or
    0 * sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[30d]))
)
/
(
  sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[30d])) > 0
)
```

PROMQL
```
# Taux d'échec sur 30 jours, par sonde et par contrôle (0 s'il n'y a eu aucun échec)
(
    sum by (job, instance, sva_check_id) (increase(sva_check_runs_total{sva_check_status="CRITICAL"}
    [30d]))
  or
    0 * sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[30d]))
)
/
(
  sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[30d])) > 0
)
```

PROMQL
```
# Disponibilité de parc sur 30 jours : toutes les sondes confondues, par contrôle
(
    sum by (sva_check_id) (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING"}[30d]))
  or
    0 * sum by (sva_check_id) (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}
    [30d]))
)
/
(
  sum by (sva_check_id) (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}
    [30d])) > 0
)
```

PROMQL
```
# Contrôles sans aucune exécution évaluable sur 30 jours (à suivre séparément)
sum by (job, instance, sva_check_id) (increase(sva_check_runs_total[30d]))
  unless on (job, instance, sva_check_id)
(sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[30d])) > 0)
```

> **Attention** — La fonction `increase()` extrapole ses résultats et ne voit pas un contrôle créé, renommé ou dont la série est apparue en cours de période comme un contrôle de 30 jours : le nombre d'exécutions obtenu peut être fractionnaire et légèrement différent du décompte exact. Pour un rapport contractuel, préférez le décompte des exécutions par l'API (§ 17.10) ou les agrégats de la plateforme d'observabilité, et vérifiez ces requêtes sur des contrôles récents et des contrôles à échecs rares avant de les utiliser. En mode envoi OTLP, les labels `job` et `instance` sont dérivés de `service.name` et `service.instance.id` (chapitre 18, § 18.5.3).

La disponibilité mesurée par la sonde est une disponibilité **observée depuis un point du réseau, à intervalles réguliers**. Une interruption plus courte que l'intervalle entre deux exécutions n'est détectée que si elle coïncide avec une exécution : si elle tombe entièrement entre deux mesures, elle passe inaperçue. Seule une interruption plus longue que l'intervalle augmenté de la gigue (`jitter`) et des éventuels retards de planification (chapitre 12) recouvre forcément au moins une exécution ; une échéance sautée ou replanifiée, par exemple après une exécution trop longue ou pendant une maintenance, allonge d'autant cet écart.

> **Information** — La détection d'une interruption plus courte que l'intervalle n'est pas garantie. Pour un engagement contractuel de niveau de service, définissez précisément la méthode de calcul (formule, traitement de `UNKNOWN` et des fenêtres de maintenance) avec les parties prenantes.

17.8 Historique, rétention et agrégats

La sonde conserve ses résultats selon trois niveaux de détail. Les durées par défaut sont modifiables par un administrateur (chapitre 9).

Données	Rétention par défaut	Contenu
Exécutions détaillées	14 jours	Résultat complet de chaque exécution, étapes, mesures, erreurs
Agrégats horaires	400 jours	Par contrôle et par heure : nombre d'exécutions, nombre d'échecs, p50 et p95 de la durée
Artefacts	7 jours et 500 Mo	Captures, journaux de console, extraits réseau

Au-delà de 14 jours, le détail d'une exécution n'est plus disponible, mais les agrégats horaires permettent d'analyser les tendances sur plus d'un an : évolution de la disponibilité, dégradation progressive des temps de réponse.

La purge des données expirées s'effectue automatiquement, par petits lots, sans interrompre l'exécution des contrôles. Si l'espace de stockage vient à manquer, la sonde purge par anticipation les données les plus anciennes, en commençant par les artefacts, et le signale par une alerte interne (§ 17.9).

✓ **Conseil** — Pour conserver un historique détaillé au-delà de la rétention locale, exportez la télémétrie vers une plateforme d'observabilité (chapitre 18). C'est le moyen prévu pour l'archivage de longue durée et pour l'analyse croisée de plusieurs sondes. Augmenter fortement la rétention locale accroît l'occupation et l'usure de la carte microSD.

i **Information** — Si un SSD NVMe facultatif est installé et activé, l'historique, les agrégats et les artefacts y sont conservés plutôt que sur la microSD. Son fonctionnement et le comportement en cas de défaillance sont décrits au chapitre 8 (§ 8.12.2).

i **Information** — Les agrégats horaires sont consultables par l'API avec `GET /api/v1/checks/{id}/aggregates?from=…&to=…&step=1h`. Chaque élément de la réponse couvre une heure et indique son début (`start`), le nombre d'exécutions (`runs`), le nombre d'échecs (`failures`) et les durées médiane et au 95e centile (`p50DurationMs`, `p95DurationMs`). La réponse est paginée (`nextCursor`). Ils sont aussi présentés dans l'interface d'administration sous forme de tendances.

17.9 État de santé de la sonde

Avant de conclure qu'une application est défaillante, il faut s'assurer que la sonde elle-même fonctionne correctement. La sonde publie son propre état de santé, distinct de celui des contrôles.

17.9.1 Consultation

Dans l'interface d'administration, l'état de santé de la sonde est affiché sur le tableau de bord et détaillé dans une page dédiée. Il comprend les informations suivantes : version, édition, durée de fonctionnement, ressources, état des composants et des destinations d'export.

Par l'API, deux ressources sont disponibles :

- `GET /health` : test de vivacité, **sans authentification**. Il répond `200` si le moteur principal répond, sans aucune information sensible. Pendant l'**état de maintenance** qui précède l'activation d'une mise à jour, il répond `{"status": "MAINTENANCE"}` avec le code HTTP `503` ; la sonde devient ensuite injoignable le temps de son redémarrage (chapitre 8, § 8.14.2). Il convient à un simple test de disponibilité de la sonde par un outil tiers.
- `GET /api/v1/system` (rôle `viewer`) : état détaillé.

```bash
curl --cacert "$CA" \
     -H @"$HOME/.sva/entete-auth" \
     "$SONDE/api/v1/system"
```

Exemple de réponse (le détail des champs figure dans la description OpenAPI de la sonde) :

```json
{
  "version": "4.2.1",
  "edition": "enterprise",
  "hostname": "sonde-paris-01",
  "site": "paris-dc1",
  "uptimeSeconds": 1296000,
  "supportEndsAt": "2030-11-30T23:59:59Z",
  "components": {
    "mainEngine": "OK",
    "scenarioEngine": "OK",
    "storage": "OK",
    "network": "OK",
    "support": "OK",
    "updates": "OK"
  },
  "resources": {
    "memoryUsedPercent": 61,
    "storageUsedPercent": 34,
    "temperatureCelsius": 58,
    "throttled": false
  },
  "exporters": [
    { "id": "dynatrace", "status": "OK", "lastSuccess": "2026-11-19T09:20:00Z" }
  ]
}
```

Chaque composant prend l'un des états `OK`, `WARNING`, `CRITICAL` ou `UNKNOWN`. Quatre d'entre eux appellent une explication particulière :

Composant	Rôle	États notables
`storage`	Volume de données de la sonde : microSD et, le cas échéant, SSD NVMe facultatif	`CRITICAL` si le SSD activé est absent ou défaillant : les contrôles continuent et l'historique local est conservé sur la microSD en mode réduit (§ 17.8)
`network`	Raccordement réseau de la sonde, y compris l'authentification 802.1X lorsqu'elle est activée (chapitre 7)	Signale un échec de l'authentification 802.1X, ou un certificat client 802.1X proche de l'expiration ; en cas d'échec, la sonde renouvelle ses tentatives et affiche l'état sur la console HDMI
`support`	Période de support de sécurité de la version installée	`WARNING` après la date de fin de support : la sonde continue de fonctionner, mais ne reçoit plus de correctifs

Composant	Rôle	États notables
`updates`	Mises à jour installées sur l'emplacement inactif et en attente d'activation (chapitre 20)	`OK` : aucune mise à jour en attente, ou activation prévue dans une fenêtre de maintenance. `WARNING` : mode automatique sans aucune fenêtre définie (configuration incomplète), mise à jour en attente sans fenêtre (aucune activation automatique possible ; un administrateur l'active par `POST /api/v1/updates/apply`), ou activation reportée (`POST /api/v1/updates/postpone`). `CRITICAL` : correctif de sécurité critique (vulnérabilité activement exploitée) en attente d'activation

La réponse indique également deux échéances, sous forme de dates RFC 3339 en UTC :

- `supportEndsAt` (toujours présent) : fin de la **période de support de sécurité** de la version majeure installée. Cette période dure au moins 5 ans après la mise sur le marché de la version majeure ; la date de fin (mois et année) est indiquée à l'achat, et le champ contient le dernier instant de ce mois. Après cette date, la sonde continue de fonctionner sans nouveau correctif ;
- `maintenanceEndsAt` : échéance du **justificatif de maintenance**, ou `null` si aucun justificatif n'a été importé. Le justificatif donne accès aux nouvelles versions fonctionnelles ; son échéance n'affecte aucune fonction acquise, et les correctifs de sécurité restent disponibles pendant la période de support.

Ces deux champs figurent aussi dans la réponse de `GET /api/v1/updates` (chapitre 20). Le champ facultatif `maintenanceEndsAt` est omis de l'exemple ci-dessus, qui reste volontairement minimal.

17.9.2 Indicateurs de santé à surveiller

Indicateur	Métrique exportée	Signal d'alerte
Sonde vivante	`sva.agent.up`	Absence de signal : la sonde est arrêtée ou injoignable (équivalent de l'état `STALE` pour la sonde entière)
Retard de planification	`sva.agent.scheduler.lag`	Retard croissant : la sonde est saturée, trop de contrôles pour sa capacité
Files d'attente	`sva.agent.queue.depth`	File qui ne se vide pas, en particulier pour la classe navigateur
Lancements du navigateur	`sva.browser.launches`	Échecs de lancement : mémoire, mise à jour incomplète
Stockage	`system.filesystem.usage` (taux d'occupation du système de fichiers) ; `sva.storage.usage` (volume par zone : base, artefacts)	Occupation proche de la capacité
Température	`sva.hw.temperature`	Température élevée : refroidissement insuffisant
Bridage matériel	`sva.hw.throttled`	`undervoltage` : alimentation insuffisante, risque de corruption de la carte ; `thermal` : surchauffe
Exports	`sva.export.failures`, `sva.export.dropped`	Échecs répétés ou données perdues vers une destination
Certificat 802.1X	`sva.dot1x.certificate.expiry`	Moins de 30 jours avant l'expiration du certificat client 802.1X, si l'authentification 802.1X est activée (la sonde émet elle-même un avertissement à cette échéance)
Ressources système	`system.cpu.utilization`, `system.memory.usage`, `system.filesystem.usage`	Saturation durable

Indicateur	Métrique exportée	Signal d'alerte
Mises à jour en attente	`sva.update.pending` (attributs `sva.update.channel`, `sva.update.critical`)	Mise à jour en attente d'activation depuis longtemps ; correctif critique en attente (`sva.update.critical="true"`)
Résultat de la dernière mise à jour	`sva.update.last_result` (attribut `sva.update.result` : `in_progress`, `succeeded`, `rolled_back`, `failed` ; Prometheus : `sva_update_last_result`)	Retour arrière (`rolled_back`) ou échec (`failed`) ; sur un site redondant, vérifiez `succeeded` sur une sonde avant d'activer la suivante (chapitre 20, § 20.4.12)
Prochaine maintenance	`sva.maintenance.next_start`, `sva.maintenance.next_end` (horodatages, en secondes)	Activation annoncée en dehors des créneaux attendus ; absence de prochaine maintenance alors qu'une mise à jour est en attente
État de maintenance	`sva.agent.maintenance` (Prometheus : `sva_agent_maintenance`)	Valeur `1` : la sonde est en état de maintenance, du drainage de ses exécutions jusqu'au retour en service après l'activation d'une mise à jour. Les alertes d'absence de signal (`sva.agent.up`) émises pendant la maintenance sont attendues : leur suspension se configure dans la plateforme (chapitre 18)

> **Avertissement** — Un bridage pour cause de sous-tension (`undervoltage`) ne doit jamais être ignoré. Une alimentation insuffisante provoque des comportements erratiques et peut corrompre la carte microSD. Remplacez l'alimentation par un modèle conforme aux prérequis (chapitre 4).

Le catalogue complet des métriques et la façon de les exploiter dans une plateforme d'observabilité sont détaillés au chapitre 18. Les symptômes de dysfonctionnement de la sonde et leurs remèdes sont traités au chapitre 19.

17.10 Exporter les données

Plusieurs besoins justifient d'extraire les résultats de la sonde : rapport mensuel de disponibilité, analyse dans un tableur, archivage réglementaire.

Besoin	Moyen recommandé
Analyse continue, historique long, multi-sondes	Export de la télémétrie vers une plateforme d'observabilité (chapitre 18)
Extraction ponctuelle d'exécutions détaillées	API `GET /api/v1/runs`, avec pagination (§ 17.3.2)
Pièces justificatives d'un incident	Détail de l'exécution et artefacts (§ 17.3.3, § 17.3.4)
Sauvegarde de la configuration (pas des résultats)	Export de configuration (chapitre 9 et chapitre 20)

L'exemple suivant extrait toutes les exécutions d'un contrôle sur une période et les convertit au format CSV. Il utilise `curl` et l'outil standard `jq`, à installer sur le poste d'administration. Il s'agit d'un exemple pédagogique, à adapter à vos besoins.

```bash
#!/usr/bin/env bash
# Extraction des exécutions d'un contrôle au format CSV (exemple pédagogique)
# Prérequis : curl, jq ; variables SONDE et CA définies et fichier d'en-tête
# "$HOME/.sva/entete-auth" créé selon le § 9.3.3 (jeton hors ligne de commande)
set -euo pipefail

CHECK="boutique-connexion"
FROM="2026-11-01T00:00:00Z"
TO="2026-12-01T00:00:00Z"
CURSOR=""

echo "id,checkId,status,startedAt,durationMs,errorType"
while : ; do
  URL="$SONDE/api/v1/runs?checkId=$CHECK&from=$FROM&to=$TO&limit=500"
  [ -n "$CURSOR" ] && URL="$URL&cursor=$CURSOR"
  PAGE=$(curl -sS --fail --cacert "$CA" \
              -H @"$HOME/.sva/entete-auth" "$URL")
  # Une ligne CSV par exécution
  echo "$PAGE" | jq -r '.items[] | [.id, .checkId, .status, .startedAt, .durationMs, (.error.type //
    "")] | @csv'
  # Page suivante, ou fin
  CURSOR=$(echo "$PAGE" | jq -r '.nextCursor // empty')
  [ -z "$CURSOR" ] && break
done
```

> **Attention** — Une extraction volumineuse sollicite la sonde. Planifiez-la en dehors des périodes chargées et limitez la période demandée. Rappelez-vous que les exécutions détaillées ne sont conservées que 14 jours par défaut : pour un rapport mensuel, extrayez les données régulièrement, ou utilisez une plateforme d'observabilité.

Le script fixe la borne de fin `TO` : l'extraction porte sur une période close. Elle ne donne le même résultat à chaque reprise que si `TO` précède suffisamment l'instant d'extraction pour qu'aucune exécution commencée avant `TO` ne soit encore en cours, et à la purge par la rétention près (§ 17.3.2) : lancez-la donc après la fin de la période, avec la marge indiquée au § 17.3.2. Si le script est interrompu plus de 24 heures, le curseur expire (`410 Gone`) : relancez alors l'extraction complète. La sonde ne fournit pas d'instantané : des exécutions purgées pendant la lecture n'apparaissent pas dans le fichier produit.

> **Information** — Le curseur renvoyé par l'API est une valeur opaque, encodée en base64url : il ne contient aucun caractère réservé dans une URL et se réinjecte tel quel, sans encodage. Ne cherchez ni à l'interpréter ni à le construire.

17.11 Synthèse

Question	Où trouver la réponse
Y a-t-il un problème en ce moment ?	Tableau de bord ; `GET /api/v1/checks/{id}/status`
Quel parcours a échoué, à quelle étape, pourquoi ?	Détail de l'exécution : `error.step`, `error.type`, `error.message`, artefacts
L'échec est-il confirmé ?	Champ `attempt` ; exécution manuelle

Question	Où trouver la réponse
Depuis quand ?	Date du changement d'état ; historique ; révision du contrôle
Est-ce local au site ?	Comparaison avec les autres contrôles et les autres sondes
L'application est-elle lente ?	Durées par étape, p95, phases HTTP
Quelle disponibilité sur le mois ?	Agrégats horaires ; plateforme d'observabilité ; formule du § 17.7.3
La sonde fonctionne-t-elle correctement ?	Santé de la sonde ; `GET /api/v1/system` ; métriques internes

Chapitre 18 — Intégration aux plateformes d'observabilité

La sonde fonctionne de manière autonome : elle exécute ses contrôles, conserve ses résultats et les présente dans son interface d'administration, sans dépendre d'aucun service extérieur. Dans une organisation qui dispose déjà d'outils de supervision, il est toutefois souhaitable que les résultats de la sonde y apparaissent. Les équipes d'exploitation retrouvent ainsi les parcours applicatifs à côté des métriques d'infrastructure, reçoivent les alertes par leurs canaux habituels et conservent l'historique dans leur outil de référence.

Ce chapitre explique comment raccorder la sonde aux plateformes suivantes :

- Prometheus et l'écosystème Grafana ;
- Datadog ;
- Dynatrace ;
- un OpenTelemetry Collector déployé chez vous ;
- Nagios et Centreon ;
- un SIEM, pour les journaux d'audit.

Il présente d'abord les principes communs (§ 18.1 à § 18.4), puis une procédure par plateforme (§ 18.5 à § 18.11). Il se termine par l'exploitation des alertes, notamment pendant la maintenance des sondes (§ 18.15), et par le dépannage des exports (§ 18.16).

> **Information** — Les adresses, identifiants d'environnement et clés figurant dans ce chapitre sont fictifs. `abc12345` désigne un identifiant d'environnement Dynatrace fictif. Les noms de domaine en `example.com` et `example.internal` sont réservés à la documentation. Les fonctions des plateformes tierces citées ici reflètent leur documentation officielle à la date de rédaction. Ces offres évoluent : vérifiez les points signalés dans la documentation de l'éditeur concerné avant toute mise en production.

18.1 Une approche fondée sur les standards ouverts

18.1.1 Pourquoi OpenTelemetry

OpenTelemetry est un standard ouvert, porté par la Cloud Native Computing Foundation, qui définit comment produire et transmettre des données de supervision. Il est pris en charge par la quasi-totalité des plateformes d'observabilité du marché. **OTLP** (*OpenTelemetry Protocol*) est le protocole de transmission associé.

La sonde produit toutes ses données de supervision au format OpenTelemetry et les transmet en OTLP. Ce choix évite un connecteur propriétaire par plateforme : se raccorder à Datadog, à Dynatrace ou à Grafana revient à **paramétrer une destination d'export** (adresse, authentification, quelques options), et non à installer un module spécifique.

En complément, la sonde expose ses métriques au **format Prometheus**, standard de fait de la collecte de métriques par interrogation. Pour Nagios et Centreon, elle fournit un plugin conforme à la convention des plugins Nagios.

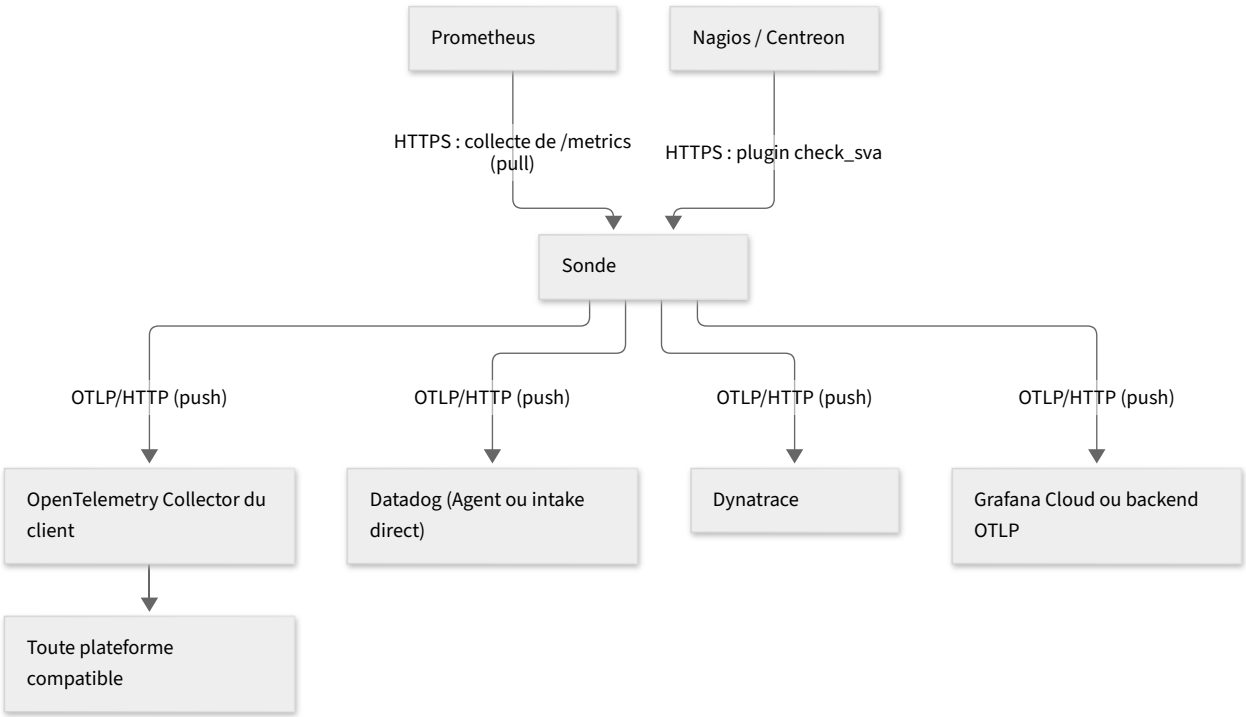


Schéma 18-1 — Envoi des données vers les plateformes d'observabilité

18.1.2 Les trois types de données

OpenTelemetry distingue trois types de données, appelés **signaux**. La sonde produit les trois.

Signal	Nature	Ce que la sonde y place	Question à laquelle il répond
Métriques	Valeurs numériques agrégées dans le temps	États des contrôles, nombre d'exécutions, durées, mesures réseau, indicateurs métier, santé de la sonde	« Combien ? À quelle fréquence ? Quelle tendance ? »
Traces	Déroulement détaillé d'une opération, découpé en **spans** (segments horodatés imbriqués)	Une trace par exécution ; un span par étape d'un scénario	« Où, dans ce parcours précis, le temps a-t-il été passé et où a-t-il échoué ? »
Logs	Événements horodatés, structurés	Fin d'exécution avec erreur, changement d'état, erreurs JavaScript des pages, sortie des plugins, opérations d'administration	« Que s'est-il passé exactement ? »

Les trois signaux portent les mêmes identifiants : identifiant du contrôle, identifiant d'exécution, identifiant de trace. Dans une plateforme qui les met en relation, on passe d'une alerte (métrique) à l'exécution concernée (trace), puis au message d'erreur (log).

18.1.3 Envoi (push) et interrogation (pull)

Deux modes de transmission coexistent :

- **Envoi (push)** : la sonde transmet périodiquement ses données à une destination. C'est le mode OTLP. La connexion est **sortante**, initiée par la sonde : aucune ouverture de port vers la sonde n'est nécessaire, ce qui convient aux plateformes SaaS.

- **Interrogation (pull)** : un outil vient lire les données sur la sonde. C'est le mode de Prometheus (lecture de `/metrics`) et de Nagios (plugin `check_sva`). La connexion est **entrante** vers la sonde, sur le port HTTPS 443, depuis le réseau d'administration.

18.1.4 OTLP/HTTP et OTLP/gRPC

OTLP existe en deux variantes de transport :

Variante	Principe	Prise en charge par la sonde	Remarques
OTLP/HTTP (encodage protobuf)	Requêtes HTTP POST vers `/v1/metrics`, `/v1/traces`, `/v1/logs`	**Mode par défaut**	Compatible avec les proxys HTTP et les pare-feu applicatifs ; seul mode accepté par Dynatrace, par l'intake direct de Datadog et par le récepteur OTLP de Prometheus
OTLP/HTTP (encodage JSON)	Idem, en JSON	Disponible	Plus volumineux ; utile surtout pour le débogage
OTLP/gRPC	Protocole gRPC sur HTTP/2	Disponible	Accepté par l'OpenTelemetry Collector et l'Agent Datadog ; plus délicat à faire traverser certains proxys

> ✓ **Conseil** — Conservez OTLP/HTTP avec l'encodage protobuf, sauf raison particulière. C'est la variante la plus largement acceptée, et elle traverse sans difficulté les proxys sortants (chapitre 7).

18.1.5 Temporalité des métriques : cumulative ou delta

Les compteurs et les histogrammes peuvent être transmis de deux manières, appelées **temporalités**. La distinction est importante, car chaque plateforme impose la sienne.

- **Cumulative** : la sonde transmet le total depuis son démarrage. Exemple : « 1 250 exécutions depuis le démarrage », puis « 1 262 exécutions » une minute plus tard. La plateforme calcule elle-même la différence.
- **Delta** : la sonde transmet uniquement l'accroissement depuis le dernier envoi. Exemple : « 12 exécutions pendant la dernière minute ».

Plateforme	Temporalité exigée
Prometheus (collecte `/metrics` ou récepteur OTLP)	Cumulative
OpenTelemetry Collector, Grafana Cloud	Cumulative (le Collector peut convertir)
Agent Datadog (récepteur OTLP)	Cumulative
Intake OTLP direct de Datadog	**Delta uniquement**
Dynatrace (API OTLP)	**Delta uniquement**

La sonde gère la temporalité **destination par destination**. Elle peut donc alimenter simultanément Prometheus en cumulatif et Dynatrace en delta, sans composant intermédiaire. Les préréglages de plateforme (§ 18.4.2) choisissent automatiquement la bonne valeur.

18.2 Ingestion n'est pas intégration

> **Attention** — Raccorder la sonde à une plateforme garantit que les données y **arrivent** (ingestion). Cela ne crée **pas automatiquement** de tableaux de bord, d'alertes ni de configuration propre à la plateforme (intégration).

Concrètement, après la configuration d'une destination :

- les métriques, traces et logs de la sonde sont disponibles dans la plateforme et interrogeables par ses outils habituels : explorateurs, langages de requête, tableaux de bord personnalisés ;
- les résultats de la sonde **n'apparaissent pas** dans les modules natifs de supervision synthétique de ces plateformes (par exemple les modules « Synthetic Monitoring » de Datadog ou de Dynatrace). Ces modules ne gèrent que les tests exécutés par leurs propres robots ;
- **aucune alerte n'est créée** : il vous appartient de définir les alertes dans la plateforme, à partir des exemples de ce chapitre ;
- **aucun tableau de bord n'est créé** : vous pouvez en construire un, ou importer les modèles fournis par l'éditeur.

Modèle fourni par l'éditeur	Format	Disponibilité
Tableau de bord Grafana	JSON d'import Grafana, source de données Prometheus	Exemple fourni avec la documentation, prêt à importer : `exemples/grafana/sva-tableau-de-bord.json` (§ 18.6.3)
Règles d'alerte Prometheus	Fichier de règles YAML (`groups:`)	Exemple fourni avec la documentation, prêt à charger : `exemples/prometheus/sva-regles-alerte.yml` (§ 18.5.5)
Mise en sourdine des fenêtres de maintenance	Fichier de configuration Alertmanager (`time_intervals` , `mute_time_intervals`)	Exemple fourni avec la documentation, à adapter à vos sondes et à leurs fenêtres : `exemples/prometheus/alertmanager-maintenance.yml` (§ 18.15.2)
Requêtes de moniteurs Datadog	Requêtes à saisir dans Datadog	Exemples de ce chapitre (§ 18.7.3) ; aucun modèle à importer
Requêtes Dynatrace	Requêtes DQL à saisir dans Dynatrace	Exemples de ce chapitre (§ 18.8.3) ; aucun modèle à importer

Les fichiers d'exemple sont livrés avec la documentation, accompagnés chacun d'une notice (`README.md`). Ils utilisent uniquement les métriques décrites au § 18.3. Leurs seuils sont indicatifs : adaptez-les à vos engagements de service avant la mise en production.

18.3 Conventions de télémétrie

Cette section décrit les données produites par la sonde. Elle constitue la référence pour construire requêtes, tableaux de bord et alertes.

> **Information** — Le préfixe `sva.` des noms de métriques et d'attributs (`sva_` au format Prometheus) est **définitif et stable** : il ne dépend pas du nom commercial du produit et ne changera pas lors d'une évolution de la marque. Vos tableaux de bord et règles d'alerte restent donc valables d'une version à l'autre. Il se distingue du préfixe `probe_` du *blackbox exporter* de Prometheus.

18.3.1 Identité de la sonde (attributs de ressource)

Chaque envoi est accompagné d'attributs qui identifient la sonde émettrice. Ces attributs ne figurent pas sur chaque point de mesure : ils décrivent l'émetteur.

Attribut	Exemple	Origine
`service.name`	`sva-agent`	Fixe
`service.version`	`4.2.1`	Version installée
`service.instance.id`	`sva-7f3c9a`	Identifiant unique généré au premier démarrage
`host.name`	`sonde-paris-01`	Nom d'hôte configuré
`host.arch`	`arm64`	Détecté
`deployment.environment.name`	`production`	Paramètre de la sonde
`sva.site`	`paris-dc1`	Paramètre de la sonde : emplacement physique ou logique
`sva.edition`	`enterprise`	Licence

> ✓ **Conseil** — Renseignez soigneusement le paramètre `site` de chaque sonde (chapitre 6). C'est lui qui permet, dans une plateforme multi-sondes, de distinguer une panne applicative (toutes les sondes en échec) d'un problème réseau local (une seule sonde en échec).

18.3.2 Catalogue des métriques

Les noms ci-dessous sont les noms OpenTelemetry. Les unités suivent la convention UCUM utilisée par OpenTelemetry : `s` pour seconde, `By` pour octet, `1` pour une grandeur sans dimension, et des annotations entre accolades comme `{run}`.

Résultats des contrôles

Métrique	Instrument	Unité	Attributs	Usage
`sva.check.runs`	Compteur	`{run}`	`sva.check.id`, `sva.check.type`, `sva.check.status`, `error.type`	Nombre d'exécutions par état ; taux d'échec ; disponibilité
`sva.check.duration`	Histogramme	`s`	`sva.check.id`, `sva.check.type`, `sva.check.status`	Temps de réponse ; percentiles
`sva.check.status`	Jauge	`{state}`	`sva.check.id`, `sva.check.type`	Dernier état : 0 = `OK`, 1 = `WARNING`, 2 = `CRITICAL`, 3 = `UNKNOWN`
`sva.check.up`	Jauge	`{state}`	`sva.check.id`, `sva.check.type`	1 si `OK` ou `WARNING`, 0 sinon
`sva.step.duration`	Histogramme	`s`	`sva.check.id`, `sva.step.name`, `sva.step.status`	Temps par étape d'un parcours
`sva.step.failures`	Compteur	`{run}`	`sva.check.id`, `sva.step.name`, `error.type`	Étape à laquelle les parcours échouent

Mesures techniques

Métrique	Instrument	Unité	Attributs	Usage
`sva.http.phase.duration`	Histogramme	`s`	`sva.check.id`, `sva.http.phase` (`dns`, `connect`, `tls`, `ttfb`, `transfer`)	Localiser une lenteur HTTP

Métrique	Instrument	Unité	Attributs	Usage
`sva.tls.certificate.expiry`	Jauge	`s`	`sva.check.id`	Durée restante avant l'expiration du certificat
`sva.icmp.rtt`	Jauge	`s`	`sva.check.id`	Temps aller-retour moyen de la dernière série
`sva.icmp.packet_loss`	Jauge	`1`	`sva.check.id`	Taux de perte, de 0 à 1
`sva.dns.lookup.duration`	Histogramme	`s`	`sva.check.id`	Durée de résolution DNS
`sva.indicator.value`	Jauge	`{value}`	`sva.check.id`, `sva.indicator.name`	Indicateurs métier issus des contrôles de base de données (chapitre 11)
`sva.plugin.perfdata`	Jauge	`{value}`	`sva.check.id`, `sva.perfdata.label`	Données de performance de plugins Nagios, déclarées exportables

Santé de la sonde

Métrique	Instrument	Unité	Attributs	Usage
`sva.agent.up`	Jauge	`{state}`	—	Battement de cœur : son absence signale une sonde arrêtée ou injoignable
`sva.agent.scheduler.lag`	Histogramme	`s`	`sva.exec.class`	Retard de démarrage des exécutions (saturation)
`sva.agent.queue.depth`	Jauge	`{job}`	`sva.exec.class`	Exécutions en attente
`sva.export.failures`	Compteur	`{batch}`	`sva.exporter.id`, `sva.signal`	Échecs d'envoi vers une destination
`sva.export.dropped`	Compteur	`{item}`	`sva.exporter.id`, `sva.signal`	Données abandonnées après échecs répétés
`sva.storage.usage`	Jauge	`By`	`sva.storage.area` (`db`, `artifacts`)	Occupation du stockage
`sva.storage.recoveries`	Compteur	`{recovery}`	—	Récupérations automatiques de la base au démarrage (chapitre 20)
`sva.browser.launches`	Compteur	`{launch}`	`sva.outcome` (`success` ou `failure`)	Fiabilité du lancement du navigateur
`sva.hw.temperature`	Jauge	`Cel`	—	Température du processeur
`sva.hw.throttled`	Jauge	`{state}`	`sva.hw.reason` (`undervoltage`, `thermal`)	Bridage pour alimentation insuffisante ou surchauffe
`sva.dot1x.certificate.expiry`	Jauge	`s`	—	Durée restante avant l'expiration du certificat client 802.1X de la sonde, si l'authentification 802.1X est activée (chapitre 7). La sonde émet elle-même un avertissement 30 jours avant l'échéance
`system.cpu.utilization`, `system.memory.usage`, `system.filesystem.usage`	Conventions OpenTelemetry	—	Selon les conventions sémantiques	Ressources système

Mises à jour et maintenance

Ces métriques annoncent les activations de mises à jour et signalent l'état de maintenance de la sonde (chapitre 20, § 20.4). Leur exploitation dans les plateformes d'alerte est décrite au § 18.15.

Métrique	Instrument	Unité	Attributs	Usage
`sva.update.pending`	Jauge	`{state}`	`sva.update.channel` (`security` ou `feature`), `sva.update.critical` (`true` ou `false`)	1 lorsqu'une mise à jour, installée sur l'emplacement système inactif, attend son activation. Un correctif critique (vulnérabilité activement exploitée) porte `sva.update.critical` = `true`
`sva.update.last_result`	Jauge	`{state}`	`sva.update.result` (`in_progress`, `succeeded`, `rolled_back` ou `failed`)	Résultat de la dernière tentative d'activation d'une mise à jour : la série dont l'attribut `sva.update.result` correspond à ce résultat vaut 1, une seule série valant 1 à la fois ; même valeur que `lastResult.outcome` de `GET /api/v1/updates`. Elle ne porte ni la version ni l'identifiant d'activation : elle ne suffit pas à valider la reprise d'une sonde après une activation (chapitre 20, § 20.4.12)
`sva.maintenance.next_start`	Jauge	`s`	—	Début de la prochaine activation planifiée, en horodatage Unix (secondes écoulées depuis le 1er janvier 1970 UTC). Valeur en l'absence de fenêtre et d'activation planifiée : 0
`sva.maintenance.next_end`	Jauge	`s`	—	Fin de la fenêtre correspondante, en horodatage Unix (secondes). Valeur en l'absence de fenêtre et d'activation planifiée : 0. Ce zéro signifie « aucune activation planifiée » ; une série absente traduit une perte de collecte
`sva.agent.maintenance`	Jauge	`{state}`	—	1 lorsque la sonde est dans l'état de maintenance (drainage puis activation), 0 sinon

18.3.3 Noms des métriques dans Prometheus

La convention de nommage de Prometheus n'utilise pas de points dans les noms (Prometheus 3 accepte des noms UTF-8, mais la traduction par défaut conserve la forme classique). Les noms OpenTelemetry y sont donc traduits : les points deviennent des tirets bas, et un suffixe d'unité ou de type est ajouté. Les principaux exemples :

Nom OpenTelemetry	Séries Prometheus correspondantes
`sva.check.runs`	`sva_check_runs_total`
`sva.check.duration`	`sva_check_duration_seconds_bucket`, `_sum`, `_count`
`sva.step.failures`	`sva_step_failures_total`
`sva.step.duration`	`sva_step_duration_seconds_bucket`, `_sum`, `_count`
`sva.http.phase.duration`	`sva_http_phase_duration_seconds_bucket`, `_sum`, `_count`
`sva.tls.certificate.expiry`	`sva_tls_certificate_expiry_seconds`
`sva.check.status`	`sva_check_status`

Nom OpenTelemetry	Séries Prometheus correspondantes
`sva.check.up`	`sva_check_up`
`sva.agent.up`	`sva_agent_up`
`sva.icmp.rtt`	`sva_icmp_rtt_seconds`
`sva.icmp.packet_loss`	`sva_icmp_packet_loss_ratio`
`sva.dns.lookup.duration`	`sva_dns_lookup_duration_seconds_bucket`, `_sum`, `_count`
`sva.indicator.value`	`sva_indicator_value`
`sva.plugin.perfdata`	`sva_plugin_perfdata`
`sva.agent.scheduler.lag`	`sva_agent_scheduler_lag_seconds_bucket`, `_sum`, `_count`
`sva.agent.queue.depth`	`sva_agent_queue_depth`
`sva.export.failures`	`sva_export_failures_total`
`sva.export.dropped`	`sva_export_dropped_total`
`sva.storage.usage`	`sva_storage_usage_bytes`
`sva.storage.recoveries`	`sva_storage_recoveries_total`
`sva.browser.launches`	`sva_browser_launches_total`
`sva.hw.temperature`	`sva_hw_temperature_celsius`
`sva.hw.throttled`	`sva_hw_throttled`
`sva.dot1x.certificate.expiry`	`sva_dot1x_certificate_expiry_seconds`
`sva.update.pending`	`sva_update_pending`
`sva.update.last_result`	`sva_update_last_result`
`sva.maintenance.next_start`	`sva_maintenance_next_start_seconds`
`sva.maintenance.next_end`	`sva_maintenance_next_end_seconds`
`sva.agent.maintenance`	`sva_agent_maintenance`

Les attributs deviennent des labels selon la même règle : `sva.check.id` devient `sva_check_id`, `sva.update.channel` devient `sva_update_channel` et `sva.update.critical` devient `sva_update_critical`.

Information — Pour une jauge d'unité `1`, la traduction Prometheus ajoute le suffixe `_ratio` (d'où `sva_icmp_packet_loss_ratio`). Les jauges d'état de la sonde sont déclarées avec l'unité annotée `{state}`, qui n'ajoute aucun suffixe : leurs noms `sva_check_status`, `sva_check_up`, `sva_agent_up`, `sva_update_pending`, `sva_update_last_result` et `sva_agent_maintenance` sont stables. De même, les indicateurs métier et les données de performance des plugins sont déclarés avec l'unité annotée `{value}`, car ils portent des valeurs de nature quelconque (un nombre de commandes, par exemple) : leurs noms `sva_indicator_value` et `sva_plugin_perfdata` ne reçoivent pas de suffixe `_ratio`. Une jauge d'unité `s` reçoit le suffixe `_seconds`, comme `sva_tls_certificate_expiry_seconds` : les horodatages de maintenance deviennent donc `sva_maintenance_next_start_seconds` et `sva_maintenance_next_end_seconds`. Leur valeur est un horodatage Unix en secondes, directement comparable à la fonction PromQL `time()`. Si votre Prometheus utilise une stratégie de traduction non standard (`translation_strategy`), vérifiez les noms sur `/metrics` (§ 18.5.2) avant d'écrire vos règles d'alerte.

18.3.4 Maîtrise de la cardinalité

La **cardinalité** est le nombre de séries temporelles distinctes produites, c'est-à-dire le nombre de combinaisons de valeurs d'attributs. Une cardinalité excessive ralentit les plateformes et, chez certains éditeurs, augmente la facturation. La sonde est conçue pour la maîtriser.

Attribut	Nombre de valeurs	Garde-fou
`sva.check.id`	Nombre de contrôles configurés	Aucun quota technique : le nombre de valeurs dépend de la seule configuration
`sva.check.type`	8	Liste fermée
`sva.check.status`	4	Liste fermée
`error.type`	22	Liste fermée (chapitre 17, § 17.5.2), dont `maintenance` pour une exécution interrompue par le drainage d'une maintenance (§ 18.15.1)
`sva.step.name`	50 au maximum par scénario	Noms d'étapes fixes, imposés par le contrat des scénarios (chapitre 14)
`sva.indicator.name`, `sva.perfdata.label`	10 au maximum par contrôle	Liste explicite dans la configuration du contrôle

Ne figurent **jamais** dans les attributs de métriques : les adresses complètes avec paramètres, les identifiants d'utilisateurs, les adresses IP résolues, les messages d'erreur, les horodatages. Ces informations, à forte cardinalité, sont portées par les traces et les logs.

Ordre de grandeur. Une sonde avec 50 contrôles, dont 10 scénarios de 8 étapes, produit environ 6 000 séries, l'essentiel provenant des histogrammes. C'est modeste pour Prometheus. Pour une plateforme qui facture les métriques personnalisées, ce volume est à estimer avec votre contrat. La sonde peut désactiver les histogrammes de durée par étape et ne conserver que les compteurs d'échecs par étape : le paramètre `metrics.stepHistograms` de la destination (§ 18.4.3), à `true` par défaut, est alors positionné à `false`.

18.3.5 Traces

Chaque exécution produit une trace :

- un **span racine** `check <identifiant du contrôle>`, qui porte l'identifiant du contrôle, son type, l'identifiant d'exécution, la révision de la configuration, le numéro de tentative et l'origine du lancement ;
- un **span par étape** `step <nom de l'étape>`, pour les scénarios et les suites de requêtes API ;
- pour les contrôles `http` et `api`, un span par requête HTTP, selon les conventions sémantiques HTTP d'OpenTelemetry. L'attribut `user_agent.synthetic.type` y vaut `test`, ce qui identifie le trafic comme synthétique.

Un span en échec porte le statut `ERROR` et un événement `exception`, qui contient le type et le message d'erreur (secrets masqués). L'identifiant de trace est également enregistré dans le résultat de l'exécution (champ `trace.traceId`, chapitre 17). On passe ainsi de la sonde à la plateforme, et réciproquement.

Les requêtes réseau individuelles d'un parcours navigateur ne sont pas tracées une à une, car le volume serait excessif. Leur synthèse est portée par le span de l'étape, et leur détail par l'artefact réseau conservé sur la sonde.

> **Information** — La sonde peut, en option, transmettre le contexte de trace (en-tête W3C `traceparent`) aux applications supervisées. Si celles-ci sont elles-mêmes instrumentées avec OpenTelemetry ou un agent APM, la trace synthétique se raccorde alors à la trace côté serveur. Cette option est désactivée par défaut. Elle s'active contrôle par contrôle et ne s'applique qu'à une liste d'origines autorisées, déclarée dans le contrôle : l'ajout d'un en-tête personnalisé à des requêtes inter-origines provoque des requêtes de contrôle CORS (*preflight*) qui peuvent perturber le fonctionnement d'une application web.

18.3.6 Logs

Événement	Sévérité	Contenu
`run.finished`	INFO si `OK` ; WARN ; ERROR	Résumé : état, durée, étape en échec, type d'erreur, message masqué, liens vers les artefacts sur la sonde
`check.status.changed`	WARN ou INFO	Transition d'état, par exemple `OK` vers `CRITICAL`
`browser.console.error`	WARN	Erreurs JavaScript émises par la page, 20 au maximum par exécution
`plugin.output`	INFO	Sortie d'un plugin Nagios, tronquée et masquée
`audit.*`	INFO	Opérations d'administration, si l'export de l'audit est activé (§ 18.11)
`agent.*`	Variable	Démarrage, récupération de base, saturation, erreurs d'export
`update.scheduled`	INFO	Annonce ou réannonce d'une activation de mise à jour. Le délai entre l'annonce et l'activation dépend du cas d'activation (matrice du chapitre 20, § 20.4.2) : au moins `updates.notice` pour une activation automatique ordinaire, au moins 15 minutes pour un correctif critique, le délai choisi pour une activation manuelle programmée, **aucun** pour une activation manuelle immédiate (émis avec `maintenance.started`). Attributs : `sva.update.activation.id` (identifiant de l'activation), `sva.update.announcement.sequence` (numéro d'annonce, strictement croissant pour la sonde), `sva.update.activation.kind` (cas d'activation : `manual`, `manual-scheduled`, `automatic`, `critical-next-window` ou `critical-asap`), `sva.update.channel` (`security` ou `feature`), `sva.update.critical` (`true` ou `false`), `sva.update.version` (version à activer), `sva.update.notice_reduced` (préavis inférieur à `updates.notice`), `sva.maintenance.start` (début prévu, ISO 8601) et, seulement s'il existe une borne de fenêtre, `sva.maintenance.end` (fin prévue, ISO 8601)
`update.cancelled`	INFO	Activation annoncée qui n'aura pas lieu sous cet identifiant : report sans instant déterminable, image remplacée, canal passé en mode `manual` ou fenêtres supprimées, image retirée. Attributs : `sva.update.activation.id`, `sva.update.announcement.sequence`, `sva.update.version`, `sva.update.cancel.reason` (`postponed`, `superseded`, `policy-changed` ou `withdrawn`)
`maintenance.started`	WARN	Entrée de la sonde dans l'état de maintenance : début du drainage, aucune nouvelle exécution ne démarre. Attributs : `sva.update.activation.id`, `sva.update.version` (version à activer)
`maintenance.ended`	INFO	Sortie de l'état de maintenance : reprise des exécutions. Attributs : `sva.update.activation.id`, `sva.update.result` (`succeeded`, `rolled_back` ou `failed`), `sva.update.version` (version active)

Ce tableau fait référence pour les attributs des événements de mise à jour et de maintenance ; leur place dans la chronologie d'une activation est décrite au § 20.4.9.

Les logs liés à une exécution portent les identifiants `trace_id` et `span_id`, ce qui permet de passer d'un log à la trace correspondante. Les captures d'écran ne sont pas transmises : elles restent sur la sonde, et le log contient l'adresse de l'artefact dans l'API de la sonde.

18.4 Configurer une destination d'export

18.4.1 Principes

Une **destination d'export** est un objet de configuration de la sonde (ressource `exporters` de l'API) qui décrit où et comment transmettre la télémétrie. La sonde peut avoir plusieurs destinations actives simultanément, par exemple Prometheus pour l'équipe infrastructure et Dynatrace pour l'équipe applicative. Chaque destination fonctionne indépendamment : une destination en panne n'affecte ni les autres, ni l'exécution des contrôles.

La configuration d'une destination requiert le rôle `admin`. Elle peut s'effectuer :

- dans l'interface d'administration, à la rubrique de gestion des destinations d'export ;
- par l'API, ressource `POST /api/v1/exporters` ;
- lors de la mise en service, dans l'assistant (chapitre 6).

18.4.2 Préréglages

Un **préréglage** (`preset`) remplit automatiquement les valeurs adaptées à une plateforme : protocole, temporalité, type d'histogramme. Les valeurs saisies explicitement priment sur celles du préréglage.

Préréglage	Usage	Protocole	Temporalité
`otlp-generic`	Tout backend OTLP, OpenTelemetry Collector	`http/protobuf` (gRPC possible)	Cumulative
`datadog-agent`	Agent Datadog présent sur votre réseau	`http/protobuf`	Cumulative
`datadog-agentless`	Intake OTLP direct de Datadog	`http/protobuf`	Delta
`dynatrace`	API OTLP de Dynatrace	`http/protobuf`	Delta
`grafana-cloud`	Passerelle OTLP de Grafana Cloud	`http/protobuf`	Cumulative
`prometheus-otlp`	Récepteur OTLP de Prometheus	`http/protobuf`, métriques uniquement	Cumulative

18.4.3 Paramètres d'une destination OTLP

Paramètre	Description	Valeur par défaut
`id`	Identifiant de la destination	—
`enabled`	Destination active	`true`
`signals`	Signaux transmis : `metrics`, `traces`, `logs`	—
`endpoint`	Adresse de base ; la sonde ajoute `/v1/metrics`, `/v1/traces`, `/v1/logs`	—
`signalEndpoints`	Adresses complètes par signal, utilisées telles quelles (sans ajout de suffixe)	—
`protocol`	`http/protobuf`, `http/json` ou `grpc`	`http/protobuf`

Paramètre	Description	Valeur par défaut
`headers`	En-têtes HTTP, notamment d'authentification. Les valeurs sensibles **doivent** référencer un secret : `${secret:NOM}`	—
`compression`	`gzip` ou `none`	`gzip`
`timeout`	Délai maximal d'un envoi	`10s`
`tls.caRef`	Autorité de certification interne à utiliser pour vérifier le serveur	Magasin de certificats système
`tls.clientCertRef`	Certificat client, pour une authentification mutuelle	—
`tls.insecureSkipVerify`	Désactive la vérification du certificat	`false` (à ne pas modifier en production)
`metrics.temporality`	`cumulative` ou `delta`	Selon le préréglage
`metrics.exportInterval`	Intervalle d'envoi des métriques	`60s`
`metrics.histogram`	`explicit` ou `exponential`	`explicit`
`metrics.stepHistograms`	Exporte les histogrammes de durée par étape ; à `false`, seuls les compteurs d'échecs par étape sont conservés (§ 18.3.4)	`true`
`retry.maxElapsed`	Durée maximale des nouvelles tentatives pour un envoi	`5m`
`queue.maxBatches`	Nombre maximal de lots en attente en mémoire ; un lot compte au plus 1 000 points	`100`
`logs.audit`	Transmet aussi les événements du journal d'audit (§ 18.11) ; nécessite le signal `logs`	`false`

Le schéma complet de l'objet est donné à l'annexe G (chapitre 21, § 21.7.4).

18.4.4 Procédure générique

Objectif : créer une destination d'export OTLP et vérifier qu'elle reçoit les données.

Prérequis :

- un jeton d'API de rôle `admin` (chapitre 9) ;
- l'adresse de la destination et ses informations d'authentification ;
- l'ouverture du flux sortant de la sonde vers la destination, directement ou via le proxy (chapitre 7).

Dans les exemples de cette procédure, le jeton d'API de la sonde n'apparaît jamais dans la ligne de commande : il est lu dans le fichier d'en-tête protégé `$HOME/.sva/entete-auth`, créé une fois selon le chapitre 9 (§ 9.3.3). Le certificat de la sonde, récupéré au format PEM et dont l'empreinte a été vérifiée selon le chapitre 6 (§ 6.4.4), est enregistré dans `sonde-ca.pem`, et la variable `SONDE` contient l'adresse de la sonde (par exemple `https://sonde-paris-01.example.internal`, nom fictif).

Étapes :

1. Enregistrez la clé ou le jeton d'authentification de la plateforme comme **secret** de la sonde. Le secret n'est plus jamais lisible ensuite, seulement remplaçable. Indiquez dans `allowedHosts` l'hôte de la destination : la sonde n'injectera ce secret que dans les requêtes adressées à cet hôte (chapitre 9, § 9.8). Exemple pour l'environnement

Dynatrace fictif du § 18.8.2, la clé étant saisie au clavier sans affichage :

```bash
read -rs VAL && printf '%s' "$VAL" \
  | jq -Rs '{value: ., allowedHosts: ["abc12345.live.dynatrace.com"]}' \
  | curl --cacert "$CA" -X PUT \
         -H @"$HOME/.sva/entete-auth" \
         -H "Content-Type: application/json" \
         -H 'If-None-Match: *' \
         --data @- \
         "$SONDE/api/v1/secrets/DT_TOKEN"; unset VAL
```

> **Information** — Le corps de la requête contient la valeur du secret, dans le champ `value`, et la liste obligatoire des hôtes autorisés, dans le champ `allowedHosts` (au moins un hôte), soit `{"value": "…", "allowedHosts": ["abc12345.live.dynatrace.com"]}`. La valeur est transmise à `curl` par son entrée standard (`--data @-`) : elle n'apparaît ni dans les arguments d'un processus, ni dans l'historique du shell. Les commandes `read` et `printf` sont internes au shell ; `jq` met la valeur au format JSON.

2. Créez la destination en référençant le secret dans l'en-tête d'authentification. Les exemples propres à chaque plateforme figurent dans les sections suivantes.

```bash
curl --cacert sonde-ca.pem -X POST \
     -H @"$HOME/.sva/entete-auth" \
     -H "Content-Type: application/json" \
     -d @destination.json \
     "$SONDE/api/v1/exporters"
```

3. Testez la destination. La sonde envoie un point de test et restitue le diagnostic : code HTTP obtenu, message d'erreur renvoyé par la plateforme, latence.

```bash
curl --cacert sonde-ca.pem -X POST \
     -H @"$HOME/.sva/entete-auth" \
     "$SONDE/api/v1/exporters/dynatrace/test"
```

   La sonde répond `202 Accepted`, avec un en-tête `Location` vers le résultat du test. Ce résultat contient les champs `status`, `httpStatus` (code HTTP renvoyé par la plateforme), `durationMs` (latence) et `message` (message d'erreur éventuel).

4. Attendez au moins deux intervalles d'export (2 minutes avec les valeurs par défaut), puis recherchez les métriques `sva.agent.up` et `sva.check.runs` dans la plateforme.

Résultat attendu : le test renvoie un succès ; la destination apparaît à l'état `OK` dans la santé de la sonde (chapitre 17, § 17.9) ; les métriques sont visibles dans la plateforme.

En cas de problème : voir le § 18.16, puis le chapitre 19 (§ 19.8). Les causes les plus fréquentes sont un flux sortant bloqué, un proxy non déclaré, une clé invalide (401 ou 403), une adresse ne correspondant pas à la région ou au site du compte, ou une temporalité refusée par la plateforme.

> **Avertissement** — N'écrivez jamais une clé d'API en clair dans l'objet de destination. Utilisez toujours la référence `${secret:NOM}`. Les exports de configuration excluent ou chiffrent les secrets, mais pas une clé saisie en clair dans un en-tête.

18.5 Prometheus

Deux modes d'intégration sont possibles. Le mode par **collecte** (pull) est recommandé : c'est le fonctionnement natif de Prometheus, et la sonde n'a rien à configurer en dehors du jeton.

18.5.1 Mode collecte (pull) : principes

Prometheus interroge périodiquement l'adresse `https://<sonde>/metrics`. La sonde y expose ses métriques au format texte Prometheus, en temporalité cumulative. L'accès est protégé :

- par HTTPS ;
- par un jeton d'API de rôle `metrics`, le rôle le plus restreint, qui ne donne accès qu'à `/metrics` et à l'état des contrôles.

> **Information** — En mode collecte, la sonde n'envoie rien. Si Prometheus ne peut pas joindre la sonde pendant un certain temps, les mesures de cette période ne sont pas récupérables dans Prometheus. Elles restent consultables sur la sonde pendant la durée de rétention (chapitre 17, § 17.8).

18.5.2 Mode collecte : procédure

Objectif : faire collecter les métriques de la sonde par un serveur Prometheus.

Prérequis :

- flux HTTPS (TCP 443) autorisé du serveur Prometheus vers la sonde ;
- certificat de la sonde, ou de l'autorité qui l'a émis, disponible sur le serveur Prometheus (chapitre 7) ;
- droits d'administration sur la configuration de Prometheus.

Étapes :

1. Créez un jeton d'API de rôle `metrics` dans l'interface d'administration de la sonde, ou par l'API (chapitre 9). Copiez-le immédiatement : il ne sera plus affiché.
2. Sur le serveur Prometheus, enregistrez le jeton dans un fichier lisible uniquement par le compte de service Prometheus, par exemple `/etc/prometheus/secrets/sonde-token`.

   ```bash
   sudo install -m 0400 -o prometheus -g prometheus /dev/null /etc/prometheus/secrets/sonde-token
   # Éditez ensuite ce fichier pour y coller le jeton, sans saut de ligne final
   ```

   Les chemins de cet exemple correspondent à une installation courante de Prometheus sous Linux ; adaptez-les à votre installation.
3. Vérifiez manuellement l'accès depuis le serveur Prometheus. Pour que le jeton n'apparaisse pas dans les arguments de `curl`, placez l'en-tête dans un fichier temporaire **unique**, créé par `mktemp` avec des droits réservés

à votre compte avant toute écriture, et effacé à la fin du test **même en cas d'échec** :

```bash
(
  set -eu
  umask 077
  entete=$(mktemp)                  # nom unique, fichier créé en mode 600
  trap 'rm -f "$entete"' EXIT       # effacement en fin de test, succès ou échec
  trap 'exit 130' INT TERM          # interruption : sortie, donc effacement
  { printf 'Authorization: Bearer '
    sudo cat /etc/prometheus/secrets/sonde-token
    printf '\n'; } > "$entete"
  curl -sS --cacert /etc/prometheus/certs/sonde-ca.pem \
       -H @"$entete" \
       https://sonde-paris-01.example.internal/metrics | head -n 20
)
```

Le jeton passe directement du fichier de Prometheus au fichier d'en-tête, créé vide en mode `600` par `mktemp` avant d'être rempli : il ne figure dans les arguments d'aucun processus, `sudo` et `cat` ne recevant que le chemin du fichier. Un nom fixe est évité, car un fichier préexistant du même nom conserverait ses propres droits. Prometheus lui-même lit le jeton dans le fichier `credentials_file` (étape 4) et n'a pas besoin de ce fichier d'en-tête.

Le résultat doit contenir des lignes commençant par `# HELP`, `# TYPE` et des séries `sva_…`. Notez les noms exacts des métriques (§ 18.3.3).

4. Ajoutez un travail de collecte (*scrape job*) dans le fichier de configuration de Prometheus :

```yaml
scrape_configs:
  - job_name: sondes
    scheme: https
    metrics_path: /metrics
    scrape_interval: 60s
    authorization:
      type: Bearer
      credentials_file: /etc/prometheus/secrets/sonde-token
    tls_config:
      ca_file: /etc/prometheus/certs/sonde-ca.pem
    static_configs:
      - targets:
          - sonde-paris-01.example.internal:443
        labels:
          site: paris-dc1
      - targets:
          - sonde-lyon-01.example.internal:443
        labels:
          site: lyon-dc1
```

5. Validez la configuration, puis rechargez Prometheus :

```bash
promtool check config /etc/prometheus/prometheus.yml
# Rechargement selon votre installation, par exemple :
sudo systemctl reload prometheus
```

Résultat attendu : dans l'interface de Prometheus, la cible du travail `sondes` apparaît à l'état `UP`, et la requête `sva_check_runs_total` renvoie des séries.

Vérification : la requête `up{job="sondes"}` vaut 1 pour chaque sonde.

> ✓ **Conseil** — Un intervalle de collecte de 60 secondes suffit : les contrôles eux-mêmes s'exécutent rarement plus d'une fois par minute. Un intervalle plus court n'apporte pas d'information supplémentaire et augmente la charge de la sonde.

> ℹ **Information** — Si plusieurs sondes partagent le même jeton, sa révocation les coupe toutes de Prometheus. Préférez un jeton par sonde, ou au minimum par site, et utilisez un fichier `credentials_file` par travail de collecte.

18.5.3 Mode envoi via le récepteur OTLP de Prometheus

Prometheus peut aussi recevoir des métriques au format OTLP. Ce mode convient lorsque la sonde n'est pas joignable par Prometheus, par exemple derrière un pare-feu qui interdit les connexions entrantes.

Contraintes à connaître (documentation officielle de Prometheus) :

- le récepteur est **désactivé par défaut** et s'active par l'option de lancement `--web.enable-otlp-receiver` ;
- il n'accepte qu'**OTLP/HTTP**, et uniquement des **métriques** ;
- l'adresse de réception est `/api/v1/otlp/v1/metrics` ;
- Prometheus n'authentifie pas **par défaut** les écritures reçues. L'authentification Basic et le TLS, y compris l'authentification des clients par certificat, se configurent dans le fichier de configuration web désigné par l'option de lancement `--web.config.file` ; ce fichier s'applique à l'ensemble de l'interface HTTP de Prometheus, et non au seul récepteur. À défaut, placez Prometheus derrière un proxy inverse qui exige une authentification, et limitez dans tous les cas l'accès réseau au récepteur.

Configuration de la destination dans la sonde. L'adresse complète est déclarée dans `signalEndpoints`, car elle ne suit pas la convention `/v1/metrics` :

```json
{
  "id": "prometheus-otlp",
  "kind": "otlp",
  "preset": "prometheus-otlp",
  "enabled": true,
  "signals": ["metrics"],
  "signalEndpoints": {
    "metrics": "https://prometheus.example.internal/api/v1/otlp/v1/metrics"
  },
  "protocol": "http/protobuf",
  "headers": { "Authorization": "Bearer ${secret:PROM_OTLP_TOKEN}" },
  "metrics": { "temporality": "cumulative", "exportInterval": "60s" }
}
```

Dans cet exemple, l'en-tête d'authentification est vérifié par le proxy inverse placé devant Prometheus, et non par Prometheus lui-même. Si vous utilisez plutôt l'authentification Basic de Prometheus (`--web.config.file`), l'en-tête prend la forme `Basic ${secret:…}`, comme pour Grafana Cloud (§ 18.6.2).

> ! **Attention** — Dans ce mode, les attributs de ressource qui identifient la sonde (§ 18.3.1) ne figurent pas sur chaque série : ils sont rattachés à une série d'information `target_info`. Seuls les labels `job` et `instance` sont présents sur toutes les séries, dérivés de `service.name` et de `service.instance.id`.

Identifier le site en mode envoi : promotion d'un attribut de ressource. En mode collecte, le label `site` est ajouté par la configuration de collecte (§ 18.5.2). En mode envoi, il n'existe pas : l'information correspondante est l'attribut de ressource `sva.site`. Pour l'utiliser dans vos requêtes et vos règles d'alerte, demandez à Prometheus de **promouvoir**

cet attribut en label de chaque série. Dans Prometheus 3, la promotion se déclare dans la section `otlp` du fichier de configuration de Prometheus, par le paramètre `promote_resource_attributes` :

```yaml
# Extrait de prometheus.yml (Prometheus 3) : promotion d'attributs de ressource OTLP en labels
otlp:
  promote_resource_attributes:
    - sva.site
```

Avec la traduction par défaut des noms (§ 18.3.3), l'attribut `sva.site` devient le label `sva_site`, présent sur toutes les séries de la sonde reçues après le rechargement de la configuration. Remplacez alors `site` par `sva_site` dans les règles d'alerte de site (§ 18.15.6). Vérifiez le nom exact du paramètre et du label dans la documentation de votre version de Prometheus, puis sur les séries reçues, avant de charger vos règles. À défaut de promotion, l'attribut reste lisible par une jointure sur la série `target_info`, ce qui alourdit chaque requête.

18.5.4 Requêtes PromQL utiles

Les requêtes ci-dessous utilisent les noms de séries du § 18.3.3, à adapter si nécessaire. Les vues **par sonde** regroupent les séries par `job`, `instance` et `sva_check_id` : deux sondes peuvent porter un contrôle de même identifiant, et un regroupement sur le seul identifiant mélangerait leurs résultats. L'**agrégation de parc**, qui additionne les exécutions de toutes les sondes, fait l'objet d'une requête distincte.

```promql
# Contrôles actuellement en CRITICAL (2) ou UNKNOWN (3), sonde par sonde
sva_check_status >= 2

# Taux d'échec de chaque contrôle sur la dernière heure, par sonde
# (exécutions évaluables : OK, WARNING ou CRITICAL ; UNKNOWN exclu)
(
    sum by (job, instance, sva_check_id) (increase(sva_check_runs_total{sva_check_status="CRITICAL"}
    [1h]))
  or
    sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[1h])) * 0
)
  /
(
  sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[1h])) > 0
)

# Disponibilité sur 30 jours, par sonde (UNKNOWN exclu, WARNING compté comme disponible)
(
    sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING"}[30d]))
  or
    sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[30d])) * 0
)
  /
(
  sum by (job, instance, sva_check_id)
    (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[30d])) > 0
)

# Agrégation de parc : taux d'échec d'un contrôle sur l'ensemble des sondes, sur la dernière heure
# (n'a de sens que si les sondes exécutent la même définition de contrôle)
(
    sum by (sva_check_id) (increase(sva_check_runs_total{sva_check_status="CRITICAL"}[1h]))
  or
    sum by (sva_check_id) (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}
    [1h])) * 0
```

```
)
  /
(
  sum by (sva_check_id) (increase(sva_check_runs_total{sva_check_status=~"OK|WARNING|CRITICAL"}[1h]))
    > 0
)

# 95e percentile du temps de réponse par contrôle et par sonde, sur 15 minutes
histogram_quantile(0.95,
  sum by (job, instance, sva_check_id, le) (rate(sva_check_duration_seconds_bucket[15m])))

# Les 5 étapes qui échouent le plus sur 24 heures, par sonde (au plus 5 lignes par couple
    job/instance)
topk by (job, instance) (5, sum by (job, instance, sva_check_id, sva_step_name)
    (increase(sva_step_failures_total[24h])))

# Certificats expirant dans moins de 21 jours
sva_tls_certificate_expiry_seconds < 21 * 86400
```

La formule de disponibilité et le traitement de l'état `UNKNOWN` sont justifiés au chapitre 17, § 17.7.3.

Ces requêtes distinguent trois situations, qui ne doivent pas être confondues :

Situation	Résultat de la requête	Mécanisme
Exécutions évaluables, sans aucun échec	`0` (taux d'échec) ou `1` (disponibilité)	Le terme `or … * 0` fournit un numérateur nul lorsque la série `CRITICAL` (ou `OK` / `WARNING`) n'a jamais été émise, par exemple pour un contrôle récent qui n'a jamais échoué : sans lui, la division ne renverrait rien
Aucune exécution évaluable sur la période (dénominateur nul : contrôle uniquement en `UNKNOWN`, ou sans exécution)	Aucun résultat pour ce contrôle	Le filtre `> 0` sur le dénominateur écarte la division par zéro ; l'absence de résultat signifie « pas de mesure », et non « 0 % » ou « 100 % »
Aucune série reçue (sonde injoignable, export interrompu)	Aucun résultat	La perte de collecte se surveille par les alertes `SondeInjoignable` ou `SondeSilencieuse` (§ 18.5.5)

Les états sont désignés explicitement (`sva_check_status=~"OK|WARNING|CRITICAL"`) plutôt que par exclusion (`!="UNKNOWN"`) : un filtre d'exclusion sélectionnerait aussi une série dépourvue du label d'état. Ces requêtes ont été vérifiées avec `promtool test rules` sur des séries simulées couvrant deux sondes homonymes, un contrôle sans échec, un contrôle uniquement en `UNKNOWN` et une série sans label d'état ; vérifiez-les sur vos propres séries avant de les intégrer à un tableau de bord.

18.5.5 Règles d'alerte

Exemple de fichier de règles d'alerte Prometheus. Les seuils sont indicatifs. Ces règles, complétées des principales alertes de santé recommandées au § 20.7.1 (l'alerte « Mise à jour en échec » s'ajoute sur le modèle des autres, à partir de `sva_update_last_result`) et des alertes de mises à jour et de maintenance (groupe `sondes-mises-a-jour`, § 18.15.3), sont fournies prêtes à charger dans le fichier d'exemple `exemples/prometheus/sva-regles-alerte.yml` (§ 18.2).

YAML

```
groups:
  - name: sondes-supervision
    rules:
      - alert: ParcoursEnEchec
        expr: sva_check_status == 2
        for: 5m
        labels:
```

```
        severity: critical
      annotations:
        summary: "Contrôle {{ $labels.sva_check_id }} en CRITICAL"
        description: "Le contrôle {{ $labels.sva_check_id }} est en échec depuis au moins 5
minutes. Consultez le détail de la dernière exécution sur la sonde {{ $labels.instance }}."

    # Pendant le drainage d'une maintenance (documentation SYNOVEX, « Maintenance de la sonde et
    # fausses alertes »), les exécutions interrompues
    # sont closes en UNKNOWN avec le type d'erreur maintenance : la clause unless neutralise
    # l'alerte tant que la sonde signale l'état de maintenance.
    - alert: ControleNonEvaluable
      expr: sva_check_status == 3 unless on(instance) sva_agent_maintenance == 1
      for: 15m
      labels:
        severity: warning
      annotations:
        summary: "Contrôle {{ $labels.sva_check_id }} en UNKNOWN"
        description: "La sonde {{ $labels.instance }} ne parvient pas à évaluer ce contrôle.
Vérifiez l'état de la sonde."

    # Alerte « Sonde muette » (documentation SYNOVEX, « Surveillance de la santé de la sonde »),
    # mode collecte.
    # Adaptez la valeur du label job au nom de votre travail de collecte.
    - alert: SondeInjoignable
      expr: up{job="sondes"} == 0
      for: 5m
      labels:
        severity: critical
      annotations:
        summary: "Sonde {{ $labels.instance }} injoignable"
        description: "Prometheus ne parvient plus à collecter la sonde. Les contrôles qu'elle
exécute ne sont plus visibles."

    # Alerte « Sonde muette » (documentation SYNOVEX, « Surveillance de la santé de la sonde »),
    # mode envoi (OTLP).
    # Règle par sonde : une sonde qui a émis sva_agent_up au cours des dernières 24 heures,
    # mais plus depuis 5 minutes, est signalée. Au-delà de 24 heures de silence, l'alerte
    # se résout d'elle-même : la règle SondeInjoignable (mode collecte) n'a pas cette limite.
    # LIMITE : la résolution après 24 heures ne signifie pas que la sonde est revenue.
    # Tenez un inventaire externe des sondes attendues (CMDB, liste de déploiement) et
    # comparez-le aux sondes qui émettent sva_agent_up (documentation SYNOVEX, « Règles d'alerte
»).
    # La clause unless neutralise l'alerte tant que la sonde signale l'état de maintenance
    # (drainage) ; le redémarrage qui suit est couvert par la mise en sourdine d'Alertmanager
    # (alertmanager-maintenance.yml).
    - alert: SondeSilencieuse
      # Mode envoi (OTLP) : le label job est dérivé de service.name (sva-agent)
      expr: |
        (
          max by (job, instance) (max_over_time(sva_agent_up{job="sva-agent"}[1d]))
            unless on(job, instance)
          max by (job, instance) (max_over_time(sva_agent_up{job="sva-agent"}[5m]))
        )
        unless on(instance) sva_agent_maintenance == 1
      labels:
        severity: critical
      annotations:
        summary: "Sonde {{ $labels.instance }} silencieuse"
        description: "Aucune série sva_agent_up reçue de la sonde {{ $labels.instance }} depuis 5
minutes (équivalent de l'état STALE pour la sonde)."

    # Alerte « Sous-tension » (documentation SYNOVEX, « Surveillance de la santé de la sonde »).
    - alert: SondeSousTension
      expr: sva_hw_throttled{sva_hw_reason="undervoltage"} == 1
      for: 5m
      labels:
        severity: warning
```

```
        annotations:
          summary: "Alimentation insuffisante sur {{ $labels.instance }}"
          description: "Risque d'instabilité et de corruption de la carte microSD. Remplacez
    l'alimentation."

      - alert: CertificatBientotExpire
        expr: sva_tls_certificate_expiry_seconds < 14 * 86400
        labels:
          severity: warning
        annotations:
          summary: "Certificat bientôt expiré ({{ $labels.sva_check_id }})"
          description: "Le certificat présenté par la cible expire dans moins de 14 jours."

      - alert: Certificat8021XBientotExpire
        expr: sva_dot1x_certificate_expiry_seconds < 30 * 86400
        labels:
          severity: warning
        annotations:
          summary: "Certificat 802.1X de la sonde {{ $labels.instance }} bientôt expiré"
          description: "Le certificat client 802.1X de la sonde expire dans moins de 30 jours.
    Renouvelez-le avant l'échéance, faute de quoi la sonde perdra l'accès au réseau."
```

Information — En mode collecte, la règle `SondeInjoignable`, fondée sur la série `up` que Prometheus produit pour chaque cible, est le moyen le plus fiable de détecter une sonde arrêtée. La règle `SondeSilencieuse` convient au mode envoi, où la série `up` n'existe pas. Elle est évaluée **sonde par sonde** : une sonde qui a émis `sva_agent_up` au cours des dernières 24 heures, mais plus depuis 5 minutes, est signalée. Contrairement à `absent_over_time`, qui ne détecte que l'absence de **toutes** les séries correspondant au filtre, cette forme distingue les sondes ; en contrepartie, l'alerte se résout d'elle-même après 24 heures de silence. Allongez la plage `[1d]` si votre politique d'astreinte l'exige.

Attention — En mode envoi, la disparition de l'alerte `SondeSilencieuse` après 24 heures **ne signifie pas que la sonde est revenue** : Prometheus a simplement oublié qu'elle existait. Tenez donc un **inventaire externe des sondes attendues** et comparez-le régulièrement aux sondes qui émettent `sva_agent_up`.

Une sonde arrêtée pendant un week-end prolongé, ou dont l'alerte n'a pas été traitée dans la journée, cesse en effet d'être signalée par `SondeSilencieuse`. L'inventaire des sondes attendues peut être une base de gestion des configurations, une liste de déploiement ou un fichier versionné. Dans Prometheus, il peut prendre la forme d'une série d'enregistrement par sonde, comparée aux séries reçues. Exemple fictif, à adapter, de règles à ajouter dans un groupe de votre fichier de règles :

YAML

```
# Une règle par sonde attendue (nom fictif) ; instance = service.instance.id
- record: sva:sonde_attendue
  expr: vector(1)
  labels:
    instance: sonde-paris-01
- alert: SondeAttendueAbsente
  expr: sva:sonde_attendue unless on (instance) sva_agent_up
  for: 10m
  labels:
    severity: critical
  annotations:
    summary: "Sonde attendue {{ $labels.instance }} sans signal"
```

Cette alerte ne se résout pas d'elle-même tant que la sonde reste absente. Elle se déclenche aussi pendant le redémarrage d'une activation : ajoutez son nom aux alertes mises en sourdine pendant les fenêtres de la sonde (§ 18.15.2).

> **Information** — La règle d'export du groupe `sondes-sante` se nomme `ExportEchecsRepetes` (anciennement `ExportEnEchec`). Elle se déclenche lorsqu'au moins un échec d'envoi survient dans chaque fenêtre glissante de 15 minutes, pendant 15 minutes consécutives : un échec isolé ne la déclenche pas, mais des échecs intermittents rapprochés la déclenchent, même si la plupart des envois réussissent. Elle ne prouve donc pas une indisponibilité continue de la destination. Les données réellement abandonnées sont signalées par `ExportDonneesPerdues` (§ 18.12).

> **Information** — Les règles `ControleNonEvaluable` et `SondeSilencieuse` comportent la clause `unless on(instance) sva_agent_maintenance == 1`, qui les neutralise tant que la sonde signale l'état de maintenance (phase de drainage). Cette clause ne couvre pas le redémarrage qui suit, pendant lequel la sonde ne produit plus aucune série. Elle n'est pas ajoutée à `SondeInjoignable` : lorsqu'une collecte échoue, Prometheus marque aussitôt comme périmées toutes les séries de la cible, y compris `sva_agent_maintenance`, et la clause resterait sans effet. Les alertes émises pendant le redémarrage se traitent dans Alertmanager (§ 18.15.2).

Validez le fichier de règles avec `promtool check rules <fichier>` avant de le charger.

18.6 Grafana

Grafana est un outil de visualisation qui s'appuie sur des sources de données. Avec la sonde, deux architectures sont courantes.

18.6.1 Pile auto-hébergée

Composant	Rôle	Alimentation par la sonde
Prometheus ou Mimir	Stockage des métriques	Collecte `/metrics`, ou OTLP (§ 18.5)
Tempo	Stockage des traces	OTLP, généralement via un collecteur
Loki	Stockage des logs	OTLP (ingestion OTLP native dans les versions récentes de Loki), généralement via un collecteur
Grafana Alloy ou OpenTelemetry Collector	Collecteur : reçoit l'OTLP de la sonde et le répartit vers Mimir, Tempo et Loki	Destination OTLP unique de la sonde
Grafana	Tableaux de bord, alertes, navigation entre métriques, traces et logs	—

Dans cette architecture, configurez dans la sonde une seule destination `otlp-generic` qui pointe vers le collecteur (Alloy ou OpenTelemetry Collector). Le collecteur se charge de la répartition par signal. Un exemple de configuration de collecteur est donné au § 18.9.

18.6.2 Grafana Cloud

Grafana Cloud propose une passerelle OTLP qui reçoit les trois signaux. Son adresse et l'identifiant d'instance sont indiqués dans le portail Grafana Cloud, sur la fiche de configuration OpenTelemetry de votre pile. L'adresse est de la forme `https://otlp-gateway-<zone>.grafana.net/otlp`.

L'authentification est de type **Basic** : la valeur de l'en-tête est l'encodage Base64 de la chaîne `<identifiant d'instance>:<jeton>`. Le jeton est un jeton de politique d'accès disposant des droits d'écriture sur les signaux transmis.

Étapes :

1. Dans le portail Grafana Cloud, relevez l'adresse de la passerelle OTLP et l'identifiant d'instance, puis créez un jeton disposant des droits d'écriture sur les métriques, traces et logs.
2. Sur votre poste, calculez la valeur Base64. N'ajoutez pas de saut de ligne :

```bash
printf '%s' '123456:<jeton Grafana Cloud>' | base64 | tr -d '\n'
```

3. Enregistrez le résultat comme secret `GRAFANA_OTLP_BASIC` sur la sonde (§ 18.4.4, étape 1).
4. Créez la destination :

```json
{
  "id": "grafana-cloud",
  "kind": "otlp",
  "preset": "grafana-cloud",
  "enabled": true,
  "signals": ["metrics", "traces", "logs"],
  "endpoint": "https://otlp-gateway-prod-eu-west-2.grafana.net/otlp",
  "protocol": "http/protobuf",
  "headers": { "Authorization": "Basic ${secret:GRAFANA_OTLP_BASIC}" },
  "metrics": { "temporality": "cumulative" }
}
```

   L'adresse ci-dessus est un exemple : utilisez celle qui est affichée dans votre portail.

5. Testez la destination (§ 18.4.4, étape 3).

> **Information** — Une erreur 401 provient le plus souvent d'un en-tête mal construit : jeton seul au lieu de `identifiant:jeton`, ou saut de ligne inclus dans l'encodage Base64.

18.6.3 Tableaux de bord et alertes dans Grafana

Les requêtes PromQL du § 18.5.4 s'utilisent telles quelles dans les panneaux Grafana dont la source est Prometheus ou Mimir. Organisation suggérée pour un tableau de bord « Supervision applicative » :

Rangée	Panneaux	Requête de base
Synthèse	Nombre de contrôles par état ; liste des contrôles en échec	`sva_check_status`
Disponibilité	Disponibilité par contrôle sur la période affichée (24 heures, 7 jours ou 30 jours selon le choix de période)	§ 18.5.4
Performance	p50 et p95 par contrôle ; durée par étape des parcours	`sva_check_duration_seconds_bucket`, `sva_step_duration_seconds_bucket`
Diagnostic	Étapes en échec, types d'erreur, phases HTTP	`sva_step_failures_total`, `sva_check_runs_total`, `sva_http_phase_duration_seconds_bucket`
Santé des sondes	Signal de vie, retard de planification, température, stockage, exports	Métriques `sva_agent_*`, `sva_hw_*`, `sva_export_*`

Utilisez une variable de tableau de bord sur le label `instance` pour comparer les sondes entre elles, et sur le label de site pour les regrouper par site.

Le fichier d'exemple `exemples/grafana/sva-tableau-de-bord.json` (§ 18.2) met en œuvre cette organisation. Il s'importe dans Grafana (menu **Dashboards**, **New**, **Import**) ; la source de données Prometheus est choisie à l'importation. Ses variables `job`, `instance` et `check` filtrent l'affichage par travail de collecte, par sonde et par contrôle. Le tableau de bord ne filtre pas par site, et ce choix est délibéré : les labels `job` et `instance` existent dans les deux modes d'intégration (en mode collecte, nom du travail et adresse de la cible ; en mode envoi, `service.name` et `service.instance.id`), alors que le label de site dépend de votre configuration. Il se nomme `site` lorsqu'il est ajouté par la configuration de collecte (§ 18.5.2), `sva_site` lorsqu'il résulte de la promotion de l'attribut `sva.site` en mode envoi (§ 18.5.3), et il est absent sans l'une de ces deux configurations. Pour filtrer par site, ajoutez au tableau de bord importé une variable sur le label que vous utilisez, puis ajoutez-la au filtre des requêtes. Une rangée supplémentaire suit l'expiration des certificats des cibles et du certificat client 802.1X des sondes. La rangée « Mises à jour et maintenance » affiche, pour chaque sonde, les mises à jour en attente d'activation (`sva_update_pending`), l'état de maintenance (`sva_agent_maintenance`) et la prochaine fenêtre planifiée (`sva_maintenance_next_start_seconds` et `sva_maintenance_next_end_seconds`).

Les alertes se définissent dans le module d'alerte de Grafana, avec les mêmes expressions que les règles du § 18.5.5. Lorsque Tempo et Loki sont alimentés, les logs `run.finished` et les traces des exécutions sont accessibles depuis les panneaux grâce à l'identifiant de trace.

18.7 Datadog

Datadog peut recevoir les données de la sonde de deux manières. Le choix dépend de la présence d'un Agent Datadog sur votre réseau.

Critère	Via l'Agent Datadog	Intake OTLP direct
Prérequis	Un Agent Datadog joignable par la sonde	Accès Internet sortant de la sonde vers Datadog
Signaux	Métriques, traces, logs	Métriques et logs ; traces selon la documentation de Datadog en vigueur (§ 18.7.2)
Temporalité	Cumulative	**Delta uniquement**
Clé d'API Datadog	Détenue par l'Agent	Stockée sur la sonde
Recommandation	**Recommandé** si un Agent existe	Lorsqu'aucun Agent n'est disponible

18.7.1 Via l'Agent Datadog (recommandé)

L'Agent Datadog intègre un récepteur OTLP. Ce récepteur est **désactivé par défaut** : l'Agent ne reçoit aucune donnée OTLP tant que vous ne l'avez pas activé dans sa configuration (étapes ci-dessous). Une fois activé, il écoute sur les adresses que vous déclarez, habituellement sur les ports standard d'OTLP, 4317 (gRPC) et 4318 (HTTP). La sonde envoie ses données à l'Agent, qui les transmet à Datadog avec sa propre clé d'API. Aucune clé Datadog n'est alors stockée sur la sonde.

Étapes côté Agent Datadog (d'après la documentation de Datadog) :

1. Activez le récepteur OTLP dans la configuration de l'Agent (`datadog.yaml`), section `otlp_config.receiver.protocols`, en écoutant sur une adresse joignable par la sonde. Les variables d'environnement `DD_OTLP_CONFIG_RECEIVER_PROTOCOLS_HTTP_ENDPOINT` et `DD_OTLP_CONFIG_RECEIVER_PROTOCOLS_GRPC_ENDPOINT` jouent le même rôle pour un Agent en conteneur.
2. Pour recevoir les logs de la sonde, activez explicitement la collecte des logs (`logs_enabled: true`) **et** l'ingestion OTLP des logs (`otlp_config.logs.enabled: true`). L'ingestion OTLP des logs est **désactivée par défaut** dans l'Agent, afin d'éviter une facturation inattendue.
3. Redémarrez l'Agent et vérifiez que le port 4318 est ouvert depuis le réseau de la sonde.

Exemple d'extrait de configuration de l'Agent :

```yaml
# Extrait de datadog.yaml (Agent Datadog)
logs_enabled: true
otlp_config:
  receiver:
    protocols:
      http:
        endpoint: 0.0.0.0:4318
  logs:
    enabled: true
```

> **Avertissement** — Le récepteur OTLP de l'Agent n'authentifie pas les émetteurs. Limitez par pare-feu l'accès au port 4318 aux seules adresses des sondes et des applications autorisées.

Étape côté sonde : créez la destination.

```json
{
  "id": "datadog-agent",
  "kind": "otlp",
  "preset": "datadog-agent",
  "enabled": true,
  "signals": ["metrics", "traces", "logs"],
  "endpoint": "http://datadog-agent.example.internal:4318",
  "protocol": "http/protobuf",
  "metrics": { "temporality": "cumulative" }
}
```

> **Information** — La liaison entre la sonde et l'Agent utilise ici HTTP sans chiffrement, ce qui est la configuration par défaut du récepteur OTLP de l'Agent. Elle reste acceptable sur un réseau interne maîtrisé. Pour la chiffrer, configurez TLS sur le récepteur de l'Agent selon la documentation de Datadog, puis utilisez une adresse `https://`.

18.7.2 Intake OTLP direct

L'intake OTLP de Datadog reçoit directement les données, sans Agent. Contraintes publiées par Datadog :

- authentification par l'en-tête `dd-api-key` contenant une clé d'API Datadog ;
- protocole **OTLP/HTTP protobuf** ;
- métriques en temporalité **delta uniquement** ;
- l'adresse dépend du **site Datadog** de votre organisation. Elle est de la forme `https://otlp.<site>`, à laquelle la sonde ajoute `/v1/metrics`, `/v1/traces` et `/v1/logs` : `https://otlp.datadoghq.com` (US1), `https://otlp.us3.datadoghq.com` (US3), `https://otlp.us5.datadoghq.com` (US5), `https://otlp.datadoghq.eu` (EU1), `https://otlp.ap1.datadoghq.com` (AP1), `https://otlp.ap2.datadoghq.com` (AP2), `https://otlp.uk1.datadoghq.com` (UK1), `https://otlp.ddog-gov.com` (US1-FED). Vérifiez l'adresse de votre site dans la documentation de Datadog, qui fait foi ;
- une erreur 403 signifie en général que l'adresse ne correspond pas au site du compte, ou que l'organisation n'a pas accès à l'intake.

Étapes :

1. Dans Datadog, créez une clé d'API dédiée aux sondes.
2. Enregistrez-la comme secret `DD_API_KEY` sur la sonde (§ 18.4.4).

3. Créez la destination. Exemple pour le site US1 :

```json
{
  "id": "datadog-direct",
  "kind": "otlp",
  "preset": "datadog-agentless",
  "enabled": true,
  "signals": ["metrics", "logs"],
  "endpoint": "https://otlp.datadoghq.com",
  "protocol": "http/protobuf",
  "headers": { "dd-api-key": "${secret:DD_API_KEY}" },
  "metrics": { "temporality": "delta", "histogram": "explicit" }
}
```

4. Testez la destination (§ 18.4.4, étape 3).

L'exemple ne transmet que les métriques et les logs (`"signals": ["metrics", "logs"]`). La sonde sait produire des traces, mais leur réception par l'intake OTLP direct dépend de l'offre de Datadog pour votre site et votre organisation : n'ajoutez `traces` à la liste `signals` qu'après avoir vérifié, dans la documentation de Datadog, que l'intake OTLP direct accepte les traces pour votre compte, puis contrôlé leur arrivée dans l'APM. Pour transmettre les traces sans cette vérification, passez par l'Agent Datadog (§ 18.7.1), qui accepte les trois signaux.

> **Attention** — Datadog impose des limites de taille par requête (5,1 Mio non compressés pour les métriques et les logs, 15 Mio pour les traces, selon sa documentation). La sonde découpe ses envois en lots adaptés. Si des erreurs 413 apparaissent dans les journaux d'export, signalez-les au support (annexe L, chapitre 21).

18.7.3 Exploitation dans Datadog

Dans Datadog, les métriques conservent leurs noms OpenTelemetry (avec des points), et leurs attributs deviennent des **tags**. Les modalités précises de conversion des attributs et des histogrammes dépendent de la configuration de Datadog (voir sa documentation sur le mappage des métriques OTLP).

Exemples de requêtes de moniteur, à adapter :

```
# Moniteur de métrique : parcours navigateur en échec
max(last_5m):max:sva.check.status{sva.check.type:browser} by {sva.check.id} >= 2

# Moniteur de métrique : absence de signal d'une sonde (option "notify if data is missing")
max(last_10m):max:sva.agent.up{*} by {host} < 1
```

Pour l'absence de signal, activez dans le moniteur l'option de notification en cas de données manquantes. C'est elle qui détecte une sonde arrêtée.

Les traces des exécutions sont visibles dans l'APM sous le service `sva-agent` (si les traces sont transmises), et les logs dans le Log Explorer, filtrables par `service:sva-agent`.

18.8 Dynatrace

Dynatrace reçoit les données de la sonde par son **API OTLP**. Contraintes publiées par Dynatrace :

- adresse de la forme `https://<identifiant-environnement>.live.dynatrace.com/api/v2/otlp` pour un environnement SaaS ; les chemins `/v1/metrics`, `/v1/traces` et `/v1/logs` y sont ajoutés. Pour un environnement Managed ou un ActiveGate, l'adresse diffère : consultez la documentation de Dynatrace ;
- **OTLP/HTTP protobuf uniquement** : gRPC n'est pas accepté ;

- métriques en temporalité **delta uniquement** : une requête ne contenant que des métriques cumulatives est rejetée (code 400), et une requête mixte est partiellement acceptée ;
- les histogrammes exponentiels perdent leurs intervalles (seuls minimum, maximum, somme et compte sont conservés), et les métriques de type *summary* ne sont pas prises en charge. La sonde utilise donc des **histogrammes explicites**, comme le fixe le préréglage `dynatrace`.

18.8.1 Jeton d'accès

Dynatrace propose deux types de jetons, selon le type d'environnement. Leurs étendues (*scopes*) diffèrent.

Type de jeton	En-tête	Étendues
Jeton d'accès classique (jeton d'API)	`Authorization: Api-Token <jeton>`	Étendues d'ingestion de l'API OTLP, par exemple `metrics.ingest` (métriques), `openTelemetryTrace.ingest` (traces) et `logs.ingest` (logs)
Jeton de plateforme	`Authorization: Bearer <jeton>`	Étendues d'ingestion propres aux jetons de plateforme, dont les noms diffèrent de ceux des jetons classiques : relevez-les dans la documentation de Dynatrace pour l'API OTLP de votre environnement

Ne cochez que les étendues correspondant aux signaux transmis. Pour un jeton de plateforme, l'étendue n'est effective que si le compte propriétaire du jeton dispose lui-même des droits correspondants. Vérifiez dans la documentation de Dynatrace le type de jeton recommandé pour votre environnement et les noms exacts de ses étendues : ils évoluent avec les versions de la plateforme.

18.8.2 Procédure

Objectif : transmettre métriques, traces et logs de la sonde à un environnement Dynatrace SaaS.

Prérequis : flux HTTPS sortant de la sonde vers `*.live.dynatrace.com`, directement ou via le proxy ; droits de création de jetons dans Dynatrace.

Étapes :

1. Dans Dynatrace, créez un jeton avec les étendues du § 18.8.1.
2. Enregistrez-le comme secret `DT_TOKEN` sur la sonde (§ 18.4.4, étape 1).
3. Créez la destination. Exemple avec un jeton classique :

   ```json
   {
     "id": "dynatrace",
     "kind": "otlp",
     "preset": "dynatrace",
     "enabled": true,
     "signals": ["metrics", "traces", "logs"],
     "endpoint": "https://abc12345.live.dynatrace.com/api/v2/otlp",
     "protocol": "http/protobuf",
     "headers": { "Authorization": "Api-Token ${secret:DT_TOKEN}" },
     "metrics": { "temporality": "delta", "histogram": "explicit", "exportInterval": "60s" }
   }
   ```

   Avec un jeton de plateforme, remplacez la valeur de l'en-tête par `Bearer ${secret:DT_TOKEN}`.
4. Testez la destination (§ 18.4.4, étape 3).
5. Dans Dynatrace, recherchez la métrique `sva.agent.up` dans l'explorateur de données ou un notebook.

Résultat attendu : les métriques `sva.*` apparaissent dans Dynatrace, les traces sont visibles dans l'explorateur de traces, et les logs `run.finished` dans la visionneuse de logs.

> **Information** — Dynatrace peut rejeter partiellement un envoi tout en renvoyant un succès (réponse de succès partiel). La sonde journalise ces rejets partiels dans ses journaux d'export (§ 18.16).

18.8.3 Exploitation dans Dynatrace

Les métriques conservent leurs noms OpenTelemetry, et leurs attributs deviennent des dimensions. Les alertes se définissent à partir de ces métriques, au moyen des mécanismes d'alerte de Dynatrace (événements métriques, règles d'anomalie). La syntaxe exacte des requêtes dépend du type d'environnement (Grail et DQL, ou sélecteurs de métriques de l'environnement classique).

Exemple de requête DQL indicative, à valider sur votre environnement :

```texte
timeseries statut = max(sva.check.status), by: { sva.check.id }
| filter arrayMax(statut) >= 2
```

Pour détecter l'arrêt d'une sonde, configurez une alerte sur l'absence de la métrique `sva.agent.up`.

18.9 OpenTelemetry Collector du client

Si votre organisation exploite déjà un **OpenTelemetry Collector** (le composant standard de réception, de transformation et de routage de la télémétrie), c'est la destination à privilégier. La sonde n'envoie qu'à un seul point, et c'est le Collector qui :

- route les données vers une ou plusieurs plateformes ;
- centralise les clés d'API des plateformes, qui ne sont alors pas stockées sur les sondes ;
- convertit les temporalités, par exemple en delta pour Dynatrace, grâce au processeur `cumulativetodelta` ;
- assure une **file d'attente persistante** sur disque, qui réduit les pertes de données en cas d'indisponibilité prolongée d'une plateforme, dans la limite de sa capacité et des données qu'il a effectivement reçues (§ 18.12).

Configuration de la destination dans la sonde :

```json
{
  "id": "collecteur",
  "kind": "otlp",
  "preset": "otlp-generic",
  "enabled": true,
  "signals": ["metrics", "traces", "logs"],
  "endpoint": "https://otel-collector.example.internal:4318",
  "protocol": "http/protobuf",
  "tls": { "caRef": "ac-interne" },
  "metrics": { "temporality": "cumulative" }
}
```

L'identifiant `ac-interne` désigne une autorité de certification interne importée dans la sonde (chapitre 7).

Exemple de configuration du Collector : réception OTLP, conversion delta, envoi à Dynatrace avec file persistante et copie des métriques vers Prometheus. Cet exemple utilise des composants de la distribution `otelcol-contrib` du Collector.

```yaml
receivers:
  otlp:
    protocols:
      http:
```

```
        endpoint: 0.0.0.0:4318
        tls:
          cert_file: /etc/otelcol/tls/collector.crt
          key_file: /etc/otelcol/tls/collector.key

processors:
  memory_limiter:
    check_interval: 1s
    limit_percentage: 80
    spike_limit_percentage: 20
  cumulativetodelta: {}
  batch: {}

extensions:
  file_storage:
    directory: /var/lib/otelcol/file_storage

exporters:
  otlphttp/dynatrace:
    endpoint: https://abc12345.live.dynatrace.com/api/v2/otlp
    headers:
      Authorization: "Api-Token ${env:DT_TOKEN}"
    sending_queue:
      enabled: true
      storage: file_storage
  prometheusremotewrite:
    endpoint: https://prometheus.example.internal/api/v1/write

service:
  extensions: [file_storage]
  pipelines:
    metrics/dynatrace:
      receivers: [otlp]
      processors: [memory_limiter, cumulativetodelta, batch]
      exporters: [otlphttp/dynatrace]
    metrics/prometheus:
      receivers: [otlp]
      processors: [memory_limiter, batch]
      exporters: [prometheusremotewrite]
    traces:
      receivers: [otlp]
      processors: [memory_limiter, batch]
      exporters: [otlphttp/dynatrace]
    logs:
      receivers: [otlp]
      processors: [memory_limiter, batch]
      exporters: [otlphttp/dynatrace]
```

Points d'attention :

- les composants disponibles et leurs paramètres dépendent de la distribution et de la version du Collector : référez-vous à sa documentation ;
- le processeur `cumulativetodelta` n'est appliqué qu'au pipeline vers Dynatrace, car Prometheus attend des données cumulatives ;
- l'écriture vers Prometheus par *remote write* exige que le récepteur correspondant soit activé sur Prometheus ;
- le répertoire de la file persistante doit exister et appartenir au compte de service du Collector.

✓ **Conseil** — Lorsque vous disposez d'un Collector, faites-lui porter les clés d'API des plateformes SaaS et ne configurez sur les sondes que l'adresse du Collector. La rotation des clés se fait alors en un seul endroit, et une sonde compromise ou volée ne livre aucune clé de plateforme.

18.10 Nagios et Centreon

Pour les organisations dont l'outil de supervision principal est Nagios ou un outil compatible comme Centreon, la sonde fournit un **plugin** conforme à la convention des plugins Nagios, nommé `check_sva`. Nagios (ou Centreon) exécute ce plugin à intervalles réguliers. Le plugin interroge la sonde en HTTPS et renvoie l'état d'un contrôle sous la forme attendue : un code retour et une ligne de texte avec des données de performance.

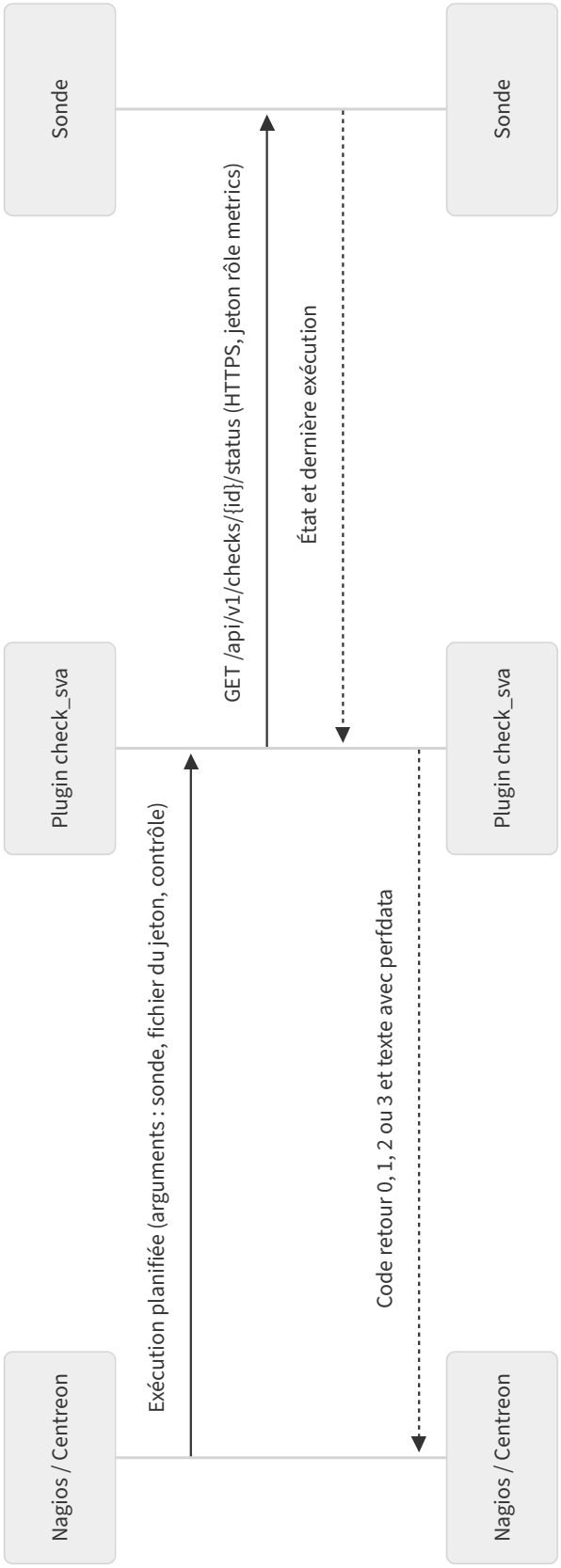


Schéma 18-2 — Interrogation d'une sonde par le plugin check_sva

Schéma présenté en orientation paysage : tournez le document d'un quart de tour pour le lire.

Ce mode fonctionne en **interrogation active** : Nagios interroge la sonde, qui n'envoie rien vers Nagios.

18.10.1 Correspondance des états

État du contrôle sur la sonde	Code retour du plugin	État Nagios
`OK`	0	OK
`WARNING`	1	WARNING
`CRITICAL`	2	CRITICAL
`UNKNOWN`	3	UNKNOWN
`STALE`	3	UNKNOWN
Sonde dans l'état de maintenance (§ 18.15.1)	3, avec une sortie de la forme `UNKNOWN - MAINTENANCE : boutique-connexion non évalué pendant l'activation d'une mise à jour`	UNKNOWN
Sonde injoignable, jeton refusé, contrôle inexistant	3	UNKNOWN

Exemple de sortie du plugin :

```
CRITICAL - boutique-connexion : étape 'Vérifier l'accès à l'espace client' (assertion_failed) |
    duration=14.65s;8;;
```

La partie qui suit le caractère `|` contient les données de performance : la durée de la dernière exécution et son seuil d'avertissement. Nagios et Centreon peuvent en tracer l'évolution.

18.10.2 Installation du plugin

Le plugin est un script Python 3 qui n'utilise que la bibliothèque standard : il ne requiert que Python 3 sur le serveur Nagios ou sur le collecteur Centreon (*poller*). Son mode de distribution est indiqué dans les notes de version. Le plugin doit être placé dans le répertoire des plugins, désigné par la macro `$USER1$` (par exemple `/usr/lib/nagios/plugins` sur une installation Debian ; adaptez à votre installation), et être exécutable par le compte de service de Nagios.

18.10.3 Syntaxe

```
check_sva -H <sonde> --token-file <fichier du jeton> -C <identifiant du contrôle> --cacert <fichier
    PEM> [-t <secondes>]
```

Option	Description
`-H`	Nom ou adresse de la sonde
`--token-file`	Fichier protégé contenant uniquement le jeton d'API de rôle `metrics`, lisible seulement par le compte qui exécute le plugin. C'est le seul mode de transmission du jeton : le plugin n'accepte pas de jeton en argument
`-C`, `--check`	Identifiant du contrôle à interroger
`--cacert`	Fichier PEM utilisé pour vérifier le certificat de la sonde : certificat autosigné de la sonde, dont l'empreinte a été vérifiée selon le chapitre 6, § 6.4.4, ou certificat de l'AC interne qui a émis le certificat de la sonde (chapitre 7, § 7.7). À toujours renseigner

Option	Description
`-t`, `--timeout`	Délai d'exécution maximal du plugin, c'est-à-dire délai d'attente de la réponse de la sonde, en secondes (10 par défaut)

Les noms d'options suivent la convention des plugins Nagios : `-t` est réservé au délai d'exécution, et `-w` et `-c` aux seuils d'avertissement et critique. `check_sva` n'utilise pas `-w` ni `-c` : l'état qu'il renvoie est celui du contrôle, évalué par la sonde avec ses propres seuils.

> **Avertissement** — Un jeton passé en argument de commande apparaîtrait dans la liste des processus du serveur de supervision, même s'il provenait d'une macro masquée dans l'interface (macro de type mot de passe de Centreon, par exemple) : la macro est remplacée par sa valeur au moment de l'exécution. C'est pourquoi le plugin ne lit le jeton que dans le fichier désigné par `--token-file` ; les macros ne contiennent que le **chemin** de ce fichier. Utilisez en outre un jeton de rôle `metrics`, qui ne permet aucune modification, et restreignez l'accès au serveur de supervision.

18.10.4 Configuration dans Nagios

Étapes :

1. Créez sur la sonde un jeton d'API de rôle `metrics` dédié à Nagios (chapitre 9).
2. Enregistrez le jeton dans un fichier lisible uniquement par le compte de Nagios, puis déclarez le chemin de ce fichier dans le fichier de ressources de Nagios (`resource.cfg`). Le chemin ci-dessous est un exemple, à adapter à votre serveur :

```
# resource.cfg : chemin du fichier contenant le jeton de la sonde (exemple)
# Le fichier contient uniquement le jeton ; droits recommandés : 0400, propriétaire nagios
$USER5$=/usr/local/nagios/etc/sonde-token
```

3. Déposez sur le serveur Nagios le certificat de la sonde, dont l'empreinte a été vérifiée selon le chapitre 6, § 6.4.4, ou celui de l'AC interne qui a émis son certificat (chapitre 7, § 7.7). Déclarez son chemin dans `resource.cfg` (exemple, à adapter) :

```
# resource.cfg : certificat utilisé par check_sva pour vérifier la sonde (exemple)
$USER6$=/usr/local/nagios/etc/sonde-paris-01.pem
```

4. Définissez la commande :

```
define command {
    command_name    check_sva
    command_line    $USER1$/check_sva -H $HOSTADDRESS$ --token-file $USER5$ --cacert $USER6$ --
    check $ARG1$
}
```

5. Déclarez la sonde comme hôte, puis un service par contrôle à suivre :

```
define host {
    use                 generic-host
    host_name           sonde-paris-01
    address             sonde-paris-01.example.internal
}

define service {
    use                 generic-service
    host_name           sonde-paris-01
    service_description Boutique - connexion
    check_command       check_sva!boutique-connexion
    check_interval      5
    retry_interval      1
    max_check_attempts  2
}
```

6. Vérifiez la configuration (`nagios -v <fichier de configuration principal>`), puis rechargez Nagios.

Résultat attendu : le service « Boutique - connexion » prend l'état du contrôle correspondant sur la sonde.

> ✓ **Conseil** — Alignez l'intervalle de vérification de Nagios sur la fréquence du contrôle dans la sonde. Interroger la sonde toutes les minutes pour un contrôle exécuté toutes les 5 minutes n'apporte aucune information nouvelle. Les nouvelles tentatives de la sonde (chapitre 17, § 17.6) confirment déjà les échecs : limitez `max_check_attempts` pour ne pas retarder inutilement l'alerte.

> ⓘ **Information** — Le modèle `generic-host` utilise par défaut une vérification par ping de l'hôte. Si le ping vers la sonde est filtré sur votre réseau, adaptez la commande de vérification de l'hôte, par exemple avec un test HTTPS de l'adresse `/health` de la sonde, qui ne nécessite pas d'authentification.

18.10.5 Configuration dans Centreon

Centreon s'appuie sur les mêmes principes. La configuration s'effectue dans son interface web :

1. Copiez le plugin sur chaque collecteur (*poller*) Centreon concerné (§ 18.10.2). Déposez aussi sur chaque collecteur le certificat de la sonde, dont l'empreinte a été vérifiée selon le chapitre 6, § 6.4.4, ou celui de l'AC interne qui a émis son certificat (chapitre 7, § 7.7), et déclarez son chemin dans une macro de ressource, ici `$USER6$`.
2. Créez sur la sonde un jeton d'API de rôle `metrics` dédié à Centreon (chapitre 9). Sur chaque collecteur concerné, enregistrez-le dans un fichier qui ne contient que le jeton, lisible uniquement par le compte qui exécute les plugins (droits recommandés : 0400). Le chemin de ce fichier est propre à votre installation.
3. Créez une **commande** de type vérification, avec la ligne de commande :

```
$USER1$/check_sva -H $HOSTADDRESS$ --token-file $_HOSTSVATOKENFILE$ --cacert $USER6$ --check
    $_SERVICECHECKID$
```

4. Sur l'hôte représentant la sonde, définissez la **macro personnalisée** `SVATOKENFILE` avec le **chemin** du fichier créé à l'étape 2. Cette macro ne contient jamais la valeur du jeton.
5. Créez un **modèle de service** utilisant cette commande, avec une macro de service `CHECKID`.
6. Créez un service par contrôle à suivre, en renseignant la macro `CHECKID` (par exemple `boutique-connexion`).
7. Exportez la configuration vers les collecteurs.

> **Avertissement** — Ne placez jamais le jeton lui-même dans une macro, même de type mot de passe : une macro masquée dans l'interface de Centreon est remplacée par sa valeur au moment de l'exécution et apparaîtrait dans la liste des processus des collecteurs (§ 18.10.3). Le plugin n'accepte d'ailleurs le jeton que par `--token-file`. Si d'anciennes configurations passent le jeton en argument, remplacez-les par la commande ci-dessus.

Les libellés exacts des menus dépendent de votre version de Centreon : référez-vous à sa documentation.

18.10.6 Quels contrôles suivre dans Nagios ou Centreon

Il n'est pas nécessaire de déclarer un service Nagios pour chaque contrôle de la sonde. Une pratique courante consiste à ne déclarer que les contrôles qui doivent déclencher une alerte dans l'outil de supervision principal (parcours critiques, API métier), et à consulter les autres dans l'interface de la sonde ou dans une plateforme d'observabilité.

> **Information** — Une fonction de découverte automatique des contrôles de la sonde par Centreon ou Nagios n'est pas prévue.

18.11 Journaux d'audit vers un SIEM

Les opérations d'administration de la sonde sont enregistrées dans son journal d'audit (chapitre 9) : connexions, modifications de configuration, changements de secrets (sans leur valeur), téléversements de scénarios, mises à jour. Pour les centraliser dans un **SIEM** (système de gestion des informations et des événements de sécurité), la sonde peut transmettre ces événements sous forme de logs OTLP, de type `audit.*`.

Mise en œuvre :

1. Activez l'export du journal d'audit sur la destination OTLP qui le recevra, avec le paramètre `logs.audit: true` (§ 18.4.3). Il est désactivé par défaut.
2. Configurez cette destination pour qu'elle transmette le signal `logs`, de préférence vers l'OpenTelemetry Collector de votre organisation. Celui-ci se charge de la transmission au SIEM, dans le format attendu par ce dernier.
3. Dans le SIEM, filtrez les événements sur `service.name = sva-agent` et les noms d'événements `audit.*`.

Le journal d'audit reste par ailleurs consultable sur la sonde (ressource `GET /api/v1/audit-events`, rôle `admin`), avec une rétention de 365 jours par défaut.

18.12 Comportement en cas d'indisponibilité d'une destination

L'indisponibilité d'une plateforme d'observabilité n'interrompt jamais la supervision. La sonde continue d'exécuter ses contrôles, d'enregistrer ses résultats et de les présenter localement.

Situation	Comportement de la sonde
Erreur temporaire : 429, 502, 503, 504, coupure réseau	Nouvelles tentatives avec un délai croissant, en respectant l'en-tête `Retry-After` s'il est fourni
Erreur définitive : 400, 401, 403, 404, 413	Abandon du lot concerné, journalisation de l'erreur, destination signalée en erreur dans la santé de la sonde
Indisponibilité au-delà de `retry.maxElapsed` (5 minutes par défaut)	Abandon des lots les plus anciens ; incrément du compteur `sva.export.dropped`
File d'attente pleine (`queue.maxBatches`)	Suppression des lots les plus anciens au profit des plus récents

Situation	Comportement de la sonde
Retour de la destination	Reprise de l'envoi des nouvelles données

Conséquences :

- **Métriques** : les points produits pendant une indisponibilité prolongée sont perdus pour la plateforme externe. Une courbe présentera un trou sur cette période. Les résultats restent intégralement consultables sur la sonde pendant la durée de rétention (chapitre 17).
- **Traces et logs** : ils suivent la même règle que les métriques.
- **Collecte Prometheus** : en mode pull, Prometheus ne récupère pas a posteriori les mesures d'une période où il n'a pas pu joindre la sonde.

→ **Évolution prévue** — Une version ultérieure permettra de rejouer, après le retour d'une destination, les traces et logs des exécutions survenues pendant l'indisponibilité, à partir de l'historique local de la sonde et dans la limite des fenêtres d'acceptation des plateformes.

✓ **Conseil** — Si votre organisation exige de **réduire les pertes** de télémétrie lors des indisponibilités de ses plateformes, faites transiter les données par un OpenTelemetry Collector disposant d'une file d'attente persistante (§ 18.9). Placez ce Collector au plus près des sondes, sur un serveur plus robuste qu'une carte microSD. Ce dispositif réduit les pertes ; il ne garantit pas la livraison, pour les raisons exposées au paragraphe « Portée d'un Collector à file persistante ».

Portée d'un Collector à file persistante. La chaîne comporte deux segments, qui ne sont pas protégés de la même manière :

Segment	Ce qui protège les données	Ce qui reste perdu
Sonde → Collector	Les nouvelles tentatives et la file en mémoire de la sonde (`retry.maxElapsed`, `queue.maxBatches`, tableau ci-dessus). Le Collector étant proche et disponible, ce segment échoue rarement	Les lots que la sonde abandonne au terme de ses reprises, ou qu'elle supprime lorsque sa file est pleine ; les lots en mémoire au moment d'un redémarrage de la sonde. Ces abandons sont comptés par `sva.export.dropped` (série `sva_export_dropped_total`, alerte `ExportDonneesPerdues`)
Collector → destination	La file persistante du Collector (`sending_queue` adossée à une extension de stockage), qui conserve les données reçues pendant l'indisponibilité de la destination	Les données au-delà de la capacité de la file ou de sa durée de rétention, celles que la destination refuse (erreur définitive, données trop anciennes pour sa fenêtre d'acceptation) et celles perdues en cas de panne du disque du Collector. Surveillez les métriques internes du Collector (taille de la file, données refusées ou abandonnées)

Le Collector ne peut préserver que ce qu'il a reçu. L'**historique local** de la sonde reste la référence en cas de doute : il conserve les résultats pendant la durée de rétention (chapitre 17), mais n'est pas renvoyé automatiquement vers les plateformes. Dimensionnez la file du Collector d'après la durée d'indisponibilité de destination que vous voulez absorber, et alertez sur `sva_export_dropped_total` comme sur les abandons du Collector.

Pour diagnostiquer une destination en erreur, utilisez le test de destination (§ 18.4.4), puis le tableau de dépannage des exports (§ 18.16).

18.12.1 Maintenance de la sonde et alertes

L'activation d'une mise à jour rend la sonde momentanément injoignable : ses données cessent d'arriver dans les plateformes, qui déclenchent alors des alertes d'absence de données alors qu'il s'agit d'une opération planifiée. La prévention de ces fausses alertes (mise en sourdine des fenêtres de maintenance, silences ponctuels, alertes propres aux mises à jour) et les alertes fondées sur l'ensemble des sondes d'un site sont décrites au § 18.15, « Exploitation des alertes ».

18.13 Sécurité des intégrations

Mesure	Détail
Secrets	Toute clé d'API ou tout jeton de plateforme est stocké comme secret de la sonde et référencé par `${secret:NOM}`. Il n'est jamais saisi en clair dans une configuration
Moindre privilège	Jetons de plateforme limités à l'ingestion des seuls signaux transmis ; jeton de rôle `metrics` pour Prometheus, Nagios et Centreon
Un jeton par usage	Un jeton distinct par outil consommateur (Prometheus, Nagios, etc.) et, si possible, par sonde, pour révoquer finement
Chiffrement	HTTPS vers les plateformes SaaS ; HTTPS pour `/metrics` et `check_sva` ; TLS recommandé vers un Collector ou un Agent interne
Vérification des certificats	Ne désactivez jamais la vérification (`insecureSkipVerify`) en production ; importez plutôt l'autorité de certification interne (chapitre 7)
Flux réseau	Seuls les flux nécessaires sont ouverts : sortant vers les destinations configurées, entrant 443 depuis les postes d'administration et, le cas échéant, Prometheus, Nagios ou Centreon (annexe E, chapitre 21)
Données transmises	Aucune valeur de secret, aucune adresse complète avec paramètres, aucun identifiant d'utilisateur dans les attributs de métriques. Les messages d'erreur transmis dans les logs et les traces sont masqués, mais peuvent contenir des informations techniques sur vos applications : choisissez des destinations dont l'hébergement et la politique de conservation sont conformes à vos exigences
Rotation	Remplacez les secrets périodiquement par l'API (`PUT /api/v1/secrets/{name}`), sans interruption des exports
Concentration des clés	Avec un Collector, les clés des plateformes SaaS restent sur le Collector, et non sur les sondes (§ 18.9)

18.14 Choisir une stratégie d'intégration

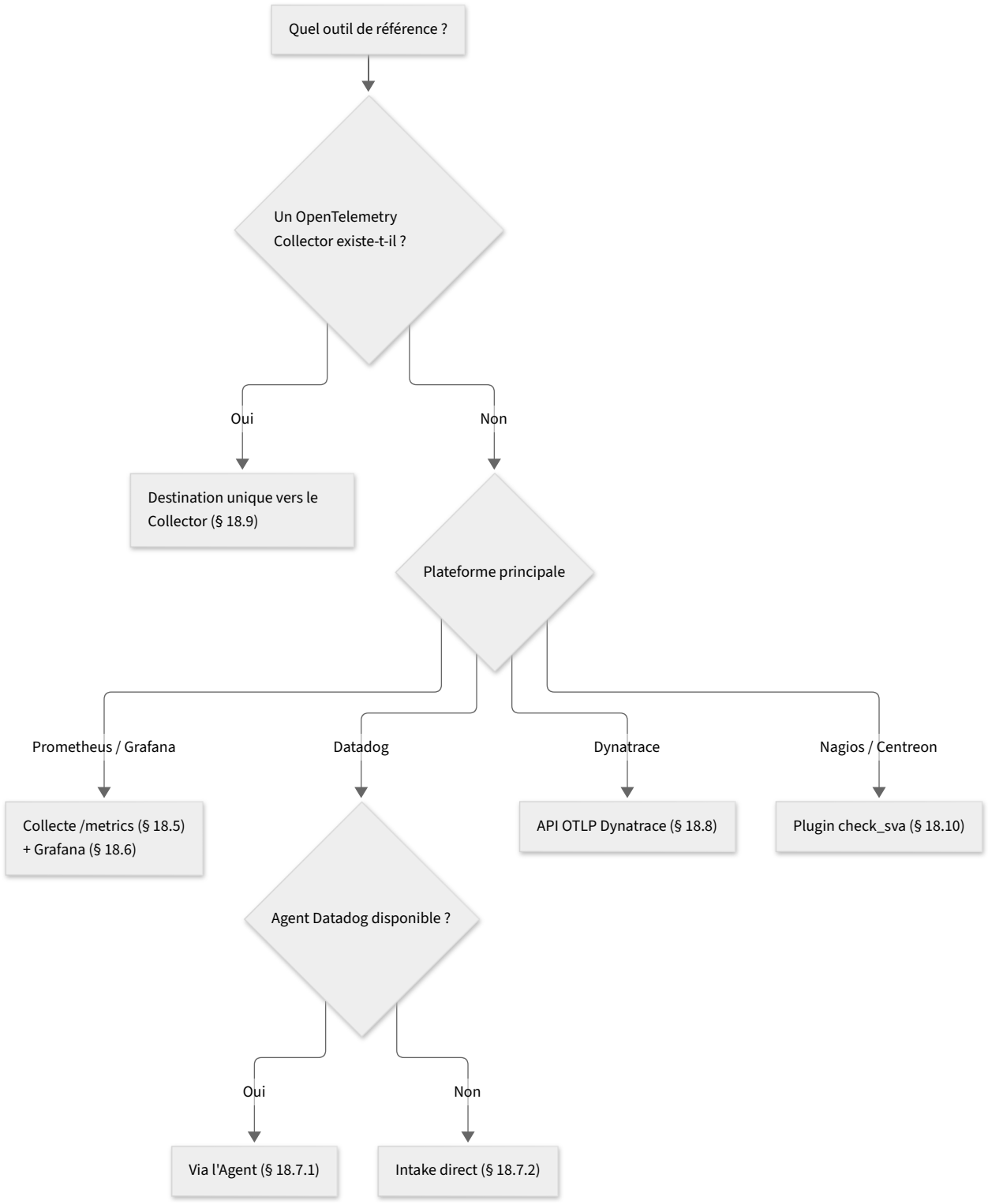


Schéma 18-3 — Choisir une stratégie d'intégration

Les modes ne sont pas exclusifs. Une configuration fréquente associe :

- le plugin `check_sva` pour les alertes dans l'outil historique de supervision ;
- l'export OTLP vers la plateforme d'observabilité, pour l'analyse, l'historique et les tableaux de bord ;
- la consultation locale de la sonde pour le diagnostic détaillé d'un échec (captures d'écran, console).

18.15 Exploitation des alertes

Cette section regroupe ce qu'il faut configurer dans vos plateformes d'alerte, une fois les intégrations en place, pour exploiter les alertes de la sonde dans la durée : éviter les fausses alertes pendant l'activation des mises à jour, traiter les activations hors fenêtre, suivre les alertes propres aux mises à jour et, sur un site équipé de plusieurs sondes, fonder les alertes sur l'ensemble des sondes du site.

18.15.1 Maintenance de la sonde et fausses alertes

L'activation d'une mise à jour, c'est-à-dire le redémarrage de la sonde sur son nouvel emplacement système, a lieu dans une **fenêtre de maintenance** définie par le client (`updates.windows`) pour une activation automatique ordinaire ou un correctif critique en mode `next-window` (réglage par défaut), à l'heure choisie par un administrateur (`POST /api/v1/updates/apply`, éventuellement avec `scheduledAt`), ou, pour un correctif critique en mode `asap` choisi explicitement, 15 minutes après son annonce, augmentées du décalage de la sonde, même hors fenêtre. La sonde n'a **aucune fenêtre par défaut** : sans fenêtre, une mise à jour ordinaire ou un correctif critique en mode `next-window` reste en attente d'activation. La matrice complète figure au chapitre 20, § 20.4.2. Le choix des fenêtres et le déroulement d'une mise à jour sont décrits au chapitre 20, § 20.4. Cette section explique comment éviter que ces activations planifiées ne déclenchent de fausses alertes dans vos plateformes, sans masquer les incidents réels.

Déroulement d'une activation vu des plateformes

Phase	Comportement de la sonde	Ce que voient les plateformes
Préavis	Selon le cas d'activation (chapitre 20, § 20.4.2) : au moins `updates.notice` (24 heures par défaut) pour une activation automatique ordinaire ; au moins 15 minutes pour un correctif critique ; aucun pour une activation manuelle immédiate (`POST /api/v1/updates/apply` sans `scheduledAt`). Un préavis inférieur à `updates.notice` est signalé par `noticeReduced: true`	Événement de log `update.scheduled` ; métriques `sva.update.pending`, `sva.maintenance.next_start` et `sva.maintenance.next_end` ; champ `scheduledActivation` de `GET /api/v1/updates`
Drainage	État de maintenance : aucune exécution nouvelle ne démarre ; celles en cours se terminent dans la limite de `updates.drainTimeout` (5 minutes par défaut, 30 minutes au maximum), puis sont interrompues en `UNKNOWN` avec le type d'erreur `maintenance`	`sva_agent_maintenance` = 1 ; événement `maintenance.started` ; `/health` répond `{"status": "MAINTENANCE"}` avec le code HTTP 503 ; `check_sva` renvoie 3 (`UNKNOWN`) avec le texte `MAINTENANCE`
Redémarrage	La sonde redémarre sur le nouvel emplacement	**Plus aucune donnée** : collecte `/metrics` en échec (`up` = 0), envois OTLP interrompus, `/health` et `check_sva` sans réponse de la sonde
Validation	Autotest de 10 minutes au plus ; en cas d'échec, retour automatique à l'emplacement précédent, avec un nouveau redémarrage	Reprise progressive des données ; éventuelle nouvelle interruption en cas de retour arrière
Reprise	Sortie de l'état de maintenance	Événement `maintenance.ended` ; `sva_agent_maintenance` = 0 ; reprise des exécutions

La durée d'indisponibilité d'une activation s'observe sur vos sondes entre les événements `maintenance.started` et `maintenance.ended` (chapitre 20, § 20.4.5) ; elle dépend notamment du délai de drainage (`updates.drainTimeout`) et de l'issue de l'autotest.

Pourquoi l'état de maintenance ne suffit pas

L'état de maintenance est exposé par la sonde elle-même. Il est donc visible pendant le drainage, mais **plus du tout pendant le redémarrage** : une sonde qui redémarre ne répond plus. Conséquences :

- en mode collecte, Prometheus constate l'échec de la collecte (`up` = 0) et marque aussitôt comme périmées toutes les séries de la sonde, y compris `sva_agent_maintenance`. Une règle qui s'appuie sur cette série ne peut plus en tenir compte ;
- en mode envoi, les séries cessent simplement d'arriver ; les règles d'absence (`SondeSilencieuse`) se déclenchent après quelques minutes ;
- Nagios et Centreon reçoivent un `UNKNOWN` « sonde injoignable », et la vérification de l'hôte fondée sur `/health` échoue ;
- les moniteurs Datadog et Dynatrace d'absence de données (`sva.agent.up`) se déclenchent.

La sonde suspend son propre calcul de l'état `STALE` pendant la maintenance, puis pendant un intervalle de planification de chaque contrôle après l'événement `maintenance.ended`, mais cette suspension n'a pas d'effet sur les plateformes externes. Il faut donc **déclarer les fenêtres de maintenance dans chaque plateforme d'alerte**, en complément des règles qui exploitent l'état de maintenance pendant le drainage.

Règles communes à toutes les plateformes

1. **Reproduisez chaque fenêtre de la sonde.** Pour chaque entrée de `updates.windows` (`days`, `start`, `duration`), déclarez une période récurrente équivalente dans la plateforme, limitée à la sonde concernée (hôte, tag ou label).
2. **Alignez les fuseaux horaires.** Les fenêtres de la sonde s'interprètent dans le fuseau horaire configuré sur la sonde (chapitre 9, § 9.9). Déclarez la période de la plateforme dans ce même fuseau, sous la forme d'un identifiant IANA (`Europe/Paris`, par exemple), et non en UTC : sinon, la période se décale d'une ou deux heures selon la saison.
3. **Prévoyez une marge après la fin de la fenêtre.** Une activation peut débuter jusqu'à la fin de la fenêtre. La période de la plateforme doit donc couvrir, au-delà de la fin de la fenêtre : le drainage (`updates.drainTimeout`), le redémarrage, l'autotest de 10 minutes et, le cas échéant, le retour arrière, qui comporte un second redémarrage. Les exemples de cette section retiennent une marge indicative de 45 minutes, à ajuster d'après la durée mesurée (chapitre 20, § 20.4) et votre valeur de `drainTimeout`.
4. **Couvrez aussi les activations hors fenêtre.** Une activation demandée par un administrateur (`POST /api/v1/updates/apply`, avec ou sans `scheduledAt`) ou un correctif critique en mode `asap` (`updates.security.criticalPolicy: asap`) peut avoir lieu en dehors des fenêtres récurrentes. Elle est annoncée par l'événement `update.scheduled` et par `sva.maintenance.next_start`, mais **sans fin prévue** et, pour une activation manuelle immédiate, **au moment même où elle commence**. Créez alors une suspension ponctuelle (silence Alertmanager, temps d'arrêt ponctuel) d'une durée maximale que vous fixez d'après la durée observée d'une activation, et levez-la après la reprise vérifiée de la sonde : procédure manuelle ou automatisation pour Alertmanager au § 18.15.4.
5. **Ne masquez que la sonde concernée.** Ne déclarez jamais une période qui suspend les alertes de toutes les sondes ou de tout l'environnement.

> ! **Attention** — Pendant la période de maintenance déclarée, les alertes de la sonde concernée sont suspendues, y compris celles qui signaleraient une panne réelle des applications supervisées. C'est la raison pour laquelle les environnements critiques doivent disposer de plusieurs sondes par site (§ 18.15.6).

Cas des correctifs de sécurité critiques. Le traitement des correctifs critiques est détaillé au § 20.4.11 ; en voici l'essentiel pour vos plateformes. Un correctif est dit critique lorsqu'il corrige une vulnérabilité activement exploitée. Il est installé dès sa publication sur l'emplacement inactif, sans interrompre les contrôles ; l'état de santé `updates` passe alors en `CRITICAL` et la métrique `sva.update.pending` porte `sva.update.critical` = `true` (alerte `CorrectifCritiqueEnAttente`, § 18.15.3). Son activation suit le paramètre `security.criticalPolicy` : en mode `next-window` (par défaut), elle a lieu au début de la première fenêtre qui commence au moins 15 minutes après l'annonce, et la mise en sourdine récurrente la couvre ; sans fenêtre définie, le correctif attend qu'un administrateur

l'active. En mode `asap`, choisi explicitement, elle a lieu 15 minutes après l'annonce, augmentées du décalage `security.criticalStagger` de la sonde (de `0` à `4h`), même hors fenêtre : elle relève alors de la suspension ponctuelle de la règle 4. Lorsque le canal `security` est en mode `manual`, aucun correctif n'est activé automatiquement. Le report est limité à 7 jours, et un administrateur peut toujours avancer l'activation. Sur un site équipé de plusieurs sondes, des décalages `security.criticalStagger` différents réduisent le risque d'activations simultanées sans l'exclure (§ 18.15.6).

18.15.2 Mise en sourdine dans Prometheus et Alertmanager

Objectif : ne pas notifier les alertes d'une sonde pendant ses fenêtres de maintenance, et neutraliser les alertes de contrôle pendant le drainage.

Prérequis : règles d'alerte des sondes chargées dans Prometheus (§ 18.5.5) ; droits d'administration sur la configuration d'Alertmanager ; fenêtres de maintenance de chaque sonde (`GET /api/v1/settings/updates`) et fuseau horaire de chaque sonde.

Étapes :

1. Chargez la version à jour du fichier `exemples/prometheus/sva-regles-alerte.yml`. Ses règles `ControleNonEvaluable` et `SondeSilencieuse` comportent la clause `unless on(instance) sva_agent_maintenance == 1`, qui couvre la phase de drainage (§ 18.5.5).
2. Dans la configuration d'Alertmanager, déclarez un **intervalle de temps** (`time_intervals`) par sonde, qui reproduit ses fenêtres augmentées de la marge, dans le fuseau horaire de la sonde (champ `location`).
3. Ajoutez, sous la route racine, une **route par sonde** qui sélectionne ses alertes de **disponibilité et d'état** (label `instance` et nom d'alerte `alertname`) et qui porte `mute_time_intervals` avec le nom de l'intervalle. La route racine ne peut pas porter `mute_time_intervals` : utilisez toujours une route enfant.
4. Placez **avant** les routes par sonde une route **sans sourdine** pour les alertes de mises à jour et de sécurité (`CorrectifCritiqueEnAttente`, `MaintenancePrologee`, alertes de certificats et d'alimentation) : ces alertes doivent être notifiées même pendant une fenêtre.
5. Validez la configuration avec `amtool check-config`, vérifiez l'acheminement avec `amtool config routes test`, puis rechargez Alertmanager.

> ! **Attention** — Une route en sourdine suspend **toutes** les alertes qu'elle sélectionne, pendant toute la durée de l'intervalle, **même si aucune activation n'a lieu** ce jour-là. Une route sélectionnée par le seul label `instance` suspendrait donc aussi un incident indépendant survenu pendant la fenêtre, un correctif critique en attente ou une maintenance qui se prolonge. Limitez la sourdine aux alertes qu'une activation déclenche normalement (`SondeInjoignable`, `SondeSilencieuse`, `ControleNonEvaluable`, `ParcoursEnEchec`, et `SondeAttendueAbsente` si vous l'utilisez), comme dans l'exemple ci-dessous. Les alertes de site (`ParcoursEnEchecSurLeSite`, `CouvertureInsuffisanteSurLeSite` et, si vous l'utilisez, `ControleAttenduNonCouvertSurLeSite`) ne portent pas de label `instance` et ne sont pas concernées. Pour une sourdine encore plus étroite, renoncez aux intervalles récurrents et créez un silence ponctuel à chaque annonce d'activation (automatisation du § 18.15.4, déclenchée par l'événement `update.scheduled`) : la sourdine ne s'applique alors qu'aux jours où une activation est réellement planifiée.

Exemple pour deux sondes d'un même site dont les fenêtres ne se chevauchent pas (fenêtres **fictives** : mardi et jeudi à 04:00 pendant 1 heure, marge de 45 minutes). Le fichier complet, avec ses commentaires, est fourni dans `exemples/prometheus/alertmanager-maintenance.yml` :

```yaml
route:
  receiver: equipe-supervision
  group_by: ['alertname', 'instance']
```

```
  routes:
    # Alertes de mises à jour et de sécurité : jamais mises en sourdine.
    # Placée en premier, cette route capte ces alertes avant les routes par sonde.
    - matchers:
        -
    alertname=~"CorrectifCritiqueEnAttente|MaintenanceProlongee|CertificatBientotExpire|Certificat802
    1XBientotExpire|SondeSousTension"
      receiver: equipe-supervision

    # Alertes de disponibilité et d'état de la sonde sonde-paris-01
    - matchers:
        - instance="sonde-paris-01.example.internal:443"
        - alertname=~"SondeInjoignable|SondeSilencieuse|ControleNonEvaluable|ParcoursEnEchec"
      receiver: equipe-supervision
      mute_time_intervals:
        - maintenance-sonde-paris-01

    # Alertes de disponibilité et d'état de la sonde sonde-paris-02
    - matchers:
        - instance="sonde-paris-02.example.internal:443"
        - alertname=~"SondeInjoignable|SondeSilencieuse|ControleNonEvaluable|ParcoursEnEchec"
      receiver: equipe-supervision
      mute_time_intervals:
        - maintenance-sonde-paris-02

    # Toutes les autres alertes suivent la route racine, sans sourdine.

time_intervals:
  - name: maintenance-sonde-paris-01
    time_intervals:
      - weekdays: ['tuesday']
        times:
          - start_time: '04:00'
            end_time: '05:45'
        # Fuseau horaire de la sonde (identifiant IANA). Sans location, Alertmanager utilise UTC.
        location: 'Europe/Paris'

  - name: maintenance-sonde-paris-02
    time_intervals:
      - weekdays: ['thursday']
        times:
          - start_time: '04:00'
            end_time: '05:45'
        location: 'Europe/Paris'

receivers:
  # Destinataire fictif : ajoutez votre intégration (courriel, messagerie, outil d'astreinte).
  - name: equipe-supervision
```

Points de syntaxe à connaître :

- dans `times`, l'heure de début est incluse et l'heure de fin exclue ; l'heure de début doit être strictement antérieure à l'heure de fin. Une fenêtre qui franchit minuit se déclare donc en deux entrées (par exemple `23:00`-`24:00` le premier jour et `00:00`-`00:45` le jour suivant) ;
- sans `location`, l'intervalle est interprété en UTC ;
- `mute_time_intervals` n'est pas hérité par les routes enfants ;
- une route en sourdine n'envoie aucune notification, mais l'alerte reste active dans Prometheus et visible dans Alertmanager ;
- les routes sont examinées dans l'ordre et la première qui correspond l'emporte : la route des alertes de mises à jour et de sécurité doit donc précéder les routes par sonde. Les alertes qui ne correspondent à aucune route enfant suivent la route racine, sans sourdine.

Le paramètre `active_time_intervals` fonctionne en sens inverse : la route ne notifie **que** pendant les intervalles listés. Il convient à une route dédiée aux alertes de maintenance que vous souhaitez recevoir uniquement pendant les fenêtres, et non à la suppression des alertes.

Résultat attendu : pendant chaque fenêtre (marge comprise), les alertes de disponibilité et d'état de la sonde concernée ne sont pas notifiées ; ses alertes de mises à jour et de sécurité, ainsi que les alertes des autres sondes et les alertes de site, le sont normalement.

Vérification :

- `amtool config routes test --config.file=<fichier> alertname=SondeInjoignable instance=<instance de la sonde>` indique le destinataire choisi pour une alerte de disponibilité de cette sonde ; répétez le test avec `alertname=CorrectifCritiqueEnAttente` pour vérifier que cette alerte suit la route sans sourdine. Avec un destinataire unique, comme dans l'exemple, attribuez temporairement un destinataire distinct à chaque route pour distinguer les routes empruntées ;
- lors d'une première activation, vérifiez dans l'interface d'Alertmanager que les alertes `SondeInjoignable` ou `SondeSilencieuse` apparaissent sans être notifiées, puis disparaissent après la reprise.

Erreurs possibles : fuseau horaire omis (intervalle en UTC) ; label `instance` différent de celui des alertes (en mode envoi, il vaut `service.instance.id`) ; fenêtre franchissant minuit déclarée en une seule entrée (refusée par `amtool check-config`) ; marge trop courte, qui laisse passer une alerte pendant l'autotest ou un retour arrière.

Précautions : une fenêtre modifiée sur la sonde doit être reportée dans Alertmanager ; vérifiez la concordance à chaque modification de `updates.windows`.

18.15.3 Alertes propres aux mises à jour

Le fichier de règles comprend un groupe `sondes-mises-a-jour`, identique à l'extrait ci-dessous :

```yaml
groups:
  - name: sondes-mises-a-jour
    rules:
      # Mise à jour installée sur l'emplacement inactif, mais aucune activation planifiée :
      # la sonde n'a pas de fenêtre de maintenance (updates.windows vide) et aucun administrateur
      # n'a programmé l'activation (POST /api/v1/updates/apply). L'état de santé updates est en
    WARNING.
      # Sans fenêtre ni activation planifiée, sva_maintenance_next_start_seconds vaut 0
    (documentation SYNOVEX,
      # « Notification, drainage et état de maintenance ») :
      # ce zéro est contractuel, et la règle l'exige explicitement (« and … == 0 »). Une série
      # absente traduit au contraire une perte de collecte, qui peut ne toucher que cette série :
      # la règle ne l'interprète jamais comme « aucune activation planifiée » et ne se déclenche
      # pas. La perte de toutes les séries de la sonde relève de SondeInjoignable ou de
      # SondeSilencieuse ; la disparition de la seule série d'horodatage n'est signalée par
      # aucune règle de ce fichier.
      - alert: MiseAJourEnAttenteSansFenetre
        expr: |
          max by (job, instance, sva_update_channel) (sva_update_pending == 1)
            and on(job, instance)
          (sva_maintenance_next_start_seconds == 0)
        for: 24h
        labels:
          severity: warning
        annotations:
          summary: "Mise à jour en attente sans activation planifiée sur {{ $labels.instance }}"
          description: "Une mise à jour du canal {{ $labels.sva_update_channel }} attend son
    activation depuis au moins 24 heures, sans fenêtre de maintenance ni activation programmée.
    Définissez une fenêtre (updates.windows) ou activez la mise à jour (POST /api/v1/updates/apply,
    éventuellement avec scheduledAt)."
```

```
      # Correctif de sécurité critique (vulnérabilité activement exploitée) en attente d'activation.
      # L'état de santé updates est en CRITICAL. Report possible de 7 jours au plus.
      - alert: CorrectifCritiqueEnAttente
        expr: max by (job, instance, sva_update_channel)
    (sva_update_pending{sva_update_critical="true"}) == 1
        for: 5m
        labels:
          severity: critical
        annotations:
          summary: "Correctif de sécurité critique en attente sur {{ $labels.instance }}"
          description: "Un correctif critique est installé sur l'emplacement inactif et attend son
    activation. Vérifiez la prochaine fenêtre de maintenance (sva_maintenance_next_start_seconds) ou
    activez-le (POST /api/v1/updates/apply). Consultez l'avis de sécurité correspondant."

      # Sonde joignable, mais dans l'état de maintenance depuis plus longtemps que prévu.
      # Seuil indicatif : drainTimeout maximal (30 minutes) et une marge de 10 minutes.
      # Pendant le redémarrage, la série disparaît : le décompte de « for » repart alors de zéro.
      - alert: MaintenancePrologee
        expr: max by (job, instance) (sva_agent_maintenance) == 1
        for: 40m
        labels:
          severity: warning
        annotations:
          summary: "Maintenance prolongée sur {{ $labels.instance }}"
          description: "La sonde {{ $labels.instance }} signale l'état de maintenance depuis plus de
    40 minutes. Aucune nouvelle exécution ne démarre pendant cet état. Consultez GET /api/v1/updates
    et le journal d'audit de la sonde."
```

Règle	Déclenchement	Action attendue
`MiseAJourEnAttenteSansFenetre`	Une mise à jour attend son activation depuis 24 heures sans fenêtre ni activation programmée	Définir une fenêtre (`updates.windows`) ou activer la mise à jour (`POST /api/v1/updates/apply`)
`CorrectifCritiqueEnAttente`	Un correctif critique attend son activation	Vérifier la date de la prochaine activation ; l'avancer si nécessaire. Le report d'un correctif critique est limité à 7 jours
`MaintenancePrologee`	La sonde est joignable mais signale l'état de maintenance depuis plus de 40 minutes	Consulter `GET /api/v1/updates` et le journal d'audit ; en cas de blocage, chapitre 19

18.15.4 Silences ponctuels pour les activations hors fenêtre

Procédure manuelle de référence. Pour une activation hors fenêtre (règle commune 4 du § 18.15.1), la procédure suivante reste sûre quelles que soient les données reçues. Elle convient seule aux activations manuelles immédiates, dont l'annonce arrive trop tard pour un automatisme.

1. Avant une activation manuelle, ou dès la réception de l'annonce dans les autres cas, créez un silence limité à la sonde et aux alertes de disponibilité et d'état :

   ```bash
   # Silence de 90 minutes au plus (durée indicative, à fixer d'après la durée observée
   # d'une activation, retour arrière compris). Adresse et labels fictifs.
   amtool silence add --alertmanager.url=https://alertmanager.example.internal \
     --duration=90m --comment="Activation manuelle sonde-paris-01" \
     job="sondes" instance="sonde-paris-01.example.internal:443" \
     alertname=~"SondeInjoignable|SondeSilencieuse|ControleNonEvaluable|ParcoursEnEchec"
   ```

2. Lancez ou laissez se dérouler l'activation, et notez l'identifiant d'activation (réponse de `POST /api/v1/updates/apply` ou attribut de l'annonce).

3. Après l'événement `maintenance.ended`, vérifiez la reprise comme au chapitre 20, § 20.4.12 : identifiant et résultat de `lastResult`, version active, `/health`, arrivée de nouveaux résultats.
4. Expirez alors le silence (`amtool silence expire <identifiant>`). Si la reprise n'est pas vérifiée, ne prolongez pas le silence : laissez-le expirer pour que les alertes reprennent, et traitez l'incident (chapitre 19, § 19.10).

Option : automatisation par les événements de mise à jour. Un automatisme de votre chaîne de traitement des logs (par exemple déclenché par les événements reçus par votre OpenTelemetry Collector ou votre SIEM) peut appliquer la même procédure. Il doit respecter les règles suivantes, faute de quoi un silence peut masquer un incident sur un mauvais créneau :

Règle	Mise en œuvre
Identifier l'activation	Chaque silence porte l'identifiant `sva.update.activation.id` et le numéro `sva.update.announcement.sequence` de l'annonce qui l'a créé
Ordonner et dédupliquer	Pour une sonde, seule l'annonce de plus grand numéro compte : une annonce ou une annulation dont le numéro ne dépasse pas le dernier traité est ignorée (doublon ou événement reçu en retard)
Sérialiser	Les événements d'une même sonde sont traités l'un après l'autre (verrou par sonde) ; deux sondes différentes sont traitées indépendamment
Remplacer sans régresser	Un nouveau silence n'expire que les silences créés pour une annonce **antérieure** de la même sonde ; il n'en touche aucun autre
Annuler	`update.cancelled` expire les silences des annonces antérieures de la sonde, sans en créer
Borner	La fin du silence est la fin annoncée (`sva.maintenance.end`) augmentée d'une marge ; sans fin annoncée, une durée maximale fixée par vous. Aucun silence n'est illimité
Lever après reprise	À `maintenance.ended`, le silence de **cette** activation n'est expiré qu'une fois la reprise constatée (sonde joignable et sortie de l'état de maintenance) ; sinon il s'achève à sa borne

Les métriques `sva_maintenance_next_start_seconds` et `sva_maintenance_next_end_seconds` ne conviennent pas à cet usage : interrogées séparément, elles peuvent refléter une autre annonce ou manquer. Prometheus sert seulement à vérifier que la sonde émettrice est connue de façon unique, puis qu'elle a repris. Exemple à adapter, pour un serveur Linux disposant de Bash, `curl`, `jq`, `flock` (paquet util-linux), `amtool` et de la commande `date` de GNU coreutils ; le fichier est fourni dans `exemples/prometheus/silence-activation.sh` :

```bash
#!/bin/bash
# Exemple : silences Alertmanager pilotés par les événements de mise à jour de la sonde.
# Appel par votre chaîne de traitement des logs, une fois par événement reçu :
#   silence-activation.sh annonce    <job> <instance> <activation.id> <sequence> <version> <début>
    [<fin>]
#   silence-activation.sh annulation <job> <instance> <activation.id> <sequence>
#   silence-activation.sh fin        <job> <instance> <activation.id> <résultat>
# annonce : update.scheduled ; annulation : update.cancelled ; fin : maintenance.ended.
# <activation.id>, <sequence>, <début>, <fin>, <résultat> : attributs sva.update.activation.id,
# sva.update.announcement.sequence, sva.maintenance.start, sva.maintenance.end (absent hors
# fenêtre : laisser vide) et sva.update.result (documentation SYNOVEX, « Logs » des
# « Conventions de télémétrie »).
# <job> et <instance> : labels Prometheus de la sonde émettrice (mode envoi : job « sva-agent »,
# instance = service.instance.id ; mode collecte : votre table de correspondance).
# Adresses fictives ; durées indicatives, à fixer d'après la durée observée d'une activation.
# Codes de sortie : 0 traité ou ignoré ; 1 arguments invalides ; 2 erreur Prometheus ;
# 3 sonde inconnue ou ambiguë ; 4 bornes incohérentes ; 5 erreur Alertmanager ;
# 6 reprise non vérifiée (réessayer plus tard) ; 7 verrou indisponible.
set -u -o pipefail

PROM='https://prometheus.example.internal'
AM='https://alertmanager.example.internal'
ETAT='/var/lib/silences-sva'   # dernier numéro d'annonce traité, par sonde
MARGE=2700                     # marge après une fin de fenêtre annoncée, en secondes
```

```
DUREE_SANS_FIN=5400            # durée maximale d'un silence sans fin annoncée, en secondes
DUREE_MAX=43200                # durée maximale plausible d'une fenêtre annoncée, en secondes
ALERTES='SondeInjoignable|SondeSilencieuse|ControleNonEvaluable|ParcoursEnEchec'

erreur() { echo "$2" >&2; exit "$1"; }
iso() { date -u -d "@$1" +%Y-%m-%dT%H:%M:%SZ; }

[ $# -ge 5 ] || erreur 1 "Usage : voir l'en-tête du script"
ACTION=$1; JOB=$2; INSTANCE=$3; ID=$4
for v in "$JOB" "$INSTANCE"; do
  [[ $v =~ ^[A-Za-z0-9._:/-]+$ ]] || erreur 1 "Valeur refusée : $v"
done
[[ $ID =~ ^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$ ]] \
  || erreur 1 "Identifiant d'activation refusé : $ID"
SONDE="$JOB/$INSTANCE"
ETIQUETTE="sva-maintenance $SONDE"

# Sérialisation par sonde : deux événements de la même sonde ne sont jamais traités en même temps.
mkdir -p "$ETAT" || erreur 7 "Dossier d'état inaccessible : $ETAT"
FICHIER="$ETAT/$(printf '%s' "$SONDE" | tr -c 'A-Za-z0-9._-' '_')"
exec 9>"$FICHIER.verrou" || erreur 7 "Verrou inaccessible"
flock -w 60 9 || erreur 7 "Verrou de $SONDE indisponible"
DERNIER=$(cat "$FICHIER.sequence" 2>/dev/null || echo 0)

# Silences actifs ou à venir créés par ce script pour cette sonde : « id sequence activation ».
silences() {
  curl -sS --fail "$AM/api/v2/silences" | jq -r --arg e "$ETIQUETTE " '.[]
    | select(.status.state != "expired" and (.comment | startswith($e)))
    | select(.comment | test("activation=[0-9a-f-]+ sequence=[0-9]+"))
    | (.comment | capture("activation=(?<a>[0-9a-f-]+) sequence=(?<s>[0-9]+)")) as $c
    | "\(.id) \($c.s) \($c.a)"'
}
expirer() {  # expirer <id>… ; un échec est signalé, jamais ignoré
  [ $# -eq 0 ] || amtool silence expire --alertmanager.url="$AM" "$@" \
    || erreur 5 "Expiration en échec : $*"
}
# Une annonce ou une annulation n'est traitée que si son numéro dépasse le dernier traité :
# les doublons et les événements reçus en retard sont ignorés.
plus_recent() {
  [[ $1 =~ ^[0-9]+$ ]] || erreur 1 "Numéro d'annonce refusé : $1"
  if [ "$1" -le "$DERNIER" ]; then
    echo "Ignoré : numéro $1 déjà dépassé pour $SONDE ($DERNIER)"; exit 0
  fi
}
# Les listes sont lues dans une variable avant usage : une erreur de lecture arrête le script.
lire() { LISTE=$(silences) || erreur 5 "Lecture des silences en échec"; }

case $ACTION in
annonce)
  [ $# -ge 7 ] || erreur 1 "annonce : <sequence> <version> <début> [<fin>] attendus"
  SEQ=$5; VERSION=$6; DEBUT_ISO=$7; FIN_ISO=${8:-}
  [[ $VERSION =~ ^[A-Za-z0-9._+-]+$ ]] || erreur 1 "Version refusée : $VERSION"
  plus_recent "$SEQ"
  MAINTENANT=$(date -u +%s)
  DEBUT=$(date -u -d "$DEBUT_ISO" +%s 2>/dev/null) || erreur 4 "Début illisible : $DEBUT_ISO"
  [ "$DEBUT" -ge "$MAINTENANT" ] || DEBUT=$MAINTENANT   # activation déjà commencée
  if [ -n "$FIN_ISO" ]; then                             # borne de fenêtre annoncée
    FIN=$(date -u -d "$FIN_ISO" +%s 2>/dev/null) || erreur 4 "Fin illisible : $FIN_ISO"
    [ "$FIN" -gt "$DEBUT" ] || erreur 4 "Fin annoncée antérieure au début"
    [ $((FIN - DEBUT)) -le "$DUREE_MAX" ] || erreur 4 "Durée annoncée invraisemblable"
    FIN=$((FIN + MARGE))
  else                                                   # aucune fin annoncée
    FIN=$((DEBUT + DUREE_SANS_FIN))
  fi
  # La sonde doit correspondre à exactement une série sva_agent_up dans Prometheus.
  R=$(curl -sS --fail --get \
    --data-urlencode "query=sva_agent_up{job=\"$JOB\",instance=\"$INSTANCE\"}" \
```

```
    "$PROM/api/v1/query") || erreur 2 "Erreur d'interrogation de Prometheus"
  [ "$(jq -r '.status' <<<"$R")" = success ] || erreur 2 "Réponse de Prometheus en erreur"
  [ "$(jq '.data.result | length' <<<"$R")" -eq 1 ] || erreur 3 "Sonde $SONDE inconnue ou ambiguë"
  lire
  # Reprise d'un traitement interrompu : un silence de ce numéro existe déjà, il n'est pas recréé.
  if ! awk -v n="$SEQ" '$2 == n { t = 1 } END { exit !t }' <<<"$LISTE"; then
    amtool silence add --alertmanager.url="$AM" \
      --start="$(iso "$DEBUT")" --end="$(iso "$FIN")" \
      --comment="$ETIQUETTE activation=$ID sequence=$SEQ version=$VERSION" \
      job="$JOB" instance="$INSTANCE" alertname=~"$ALERTES" >/dev/null \
      || erreur 5 "Création en échec"
  fi
  # N'expirer que les silences d'annonces antérieures de la même sonde, puis retenir le numéro.
  expirer $(awk -v n="$SEQ" '$2 < n { print $1 }' <<<"$LISTE")
  echo "$SEQ" >"$FICHIER.sequence"
  echo "Silence de $SONDE : activation $ID, annonce $SEQ, jusqu'à $(iso "$FIN")" ;;
annulation)
  SEQ=$5
  plus_recent "$SEQ"
  lire
  expirer $(awk -v n="$SEQ" '$2 < n { print $1 }' <<<"$LISTE")
  echo "$SEQ" >"$FICHIER.sequence"
  echo "Activation $ID annulée : silences antérieurs de $SONDE expirés" ;;
fin)
  RESULTAT=$5
  [[ $RESULTAT =~ ^(succeeded|rolled_back|failed)$ ]] || erreur 1 "Résultat refusé : $RESULTAT"
  # Reprise vérifiée : sonde joignable et sortie de l'état de maintenance.
  R=$(curl -sS --fail --get --data-urlencode \
    "query=sva_agent_up{job=\"$JOB\",instance=\"$INSTANCE\"} == 1 and on(job, instance)
   sva_agent_maintenance{job=\"$JOB\",instance=\"$INSTANCE\"} == 0" \
    "$PROM/api/v1/query") || erreur 2 "Erreur d'interrogation de Prometheus"
  [ "$(jq '.data.result | length' <<<"$R")" -eq 1 ] \
    || erreur 6 "Reprise de $SONDE non vérifiée : silence conservé jusqu'à sa fin"
  lire
  expirer $(awk -v a="$ID" '$3 == a { print $1 }' <<<"$LISTE")
  echo "Maintenance $ID terminée ($RESULTAT) : silences de cette activation expirés" ;;
*) erreur 1 "Action inconnue : $ACTION" ;;
esac
```

`amtool silence add` accepte des dates de début et de fin au format RFC 3339 (`--start`, `--end`) et des filtres sur les labels. Le script lit les silences existants par l'API d'Alertmanager (`GET /api/v2/silences`) et ne considère que ceux qu'il a créés, reconnus à leur commentaire. Le filtre sur `alertname` limite le silence aux alertes de disponibilité et d'état : les alertes de mises à jour et de sécurité de la sonde restent notifiées. Si votre Alertmanager exige une authentification, utilisez l'option `--http.config.file` d'`amtool` et un fichier d'en-tête protégé pour `curl` (`-H @fichier`), jamais un jeton en argument ; procédez de même pour Prometheus.

Limites de l'automatisation. Elle réduit le travail manuel sans le supprimer :

- **activation manuelle immédiate** : l'annonce est émise au moment où l'activation commence et peut arriver après le début du redémarrage. Appliquez la procédure manuelle de référence **avant** la commande ; l'annonce reçue ensuite est traitée normalement par l'automatisme, dont le silence s'ajoute au silence manuel. Expirez ce dernier après la reprise vérifiée, comme à l'étape 4 ;
- **événement perdu** : si une annonce n'est jamais reçue, aucun silence n'est créé et les alertes se déclenchent ; si une annulation ou une fin de maintenance est perdue, le silence s'achève à sa borne. Dans les deux cas, l'erreur va dans le sens de la notification ;
- **ordre des annonces** : il repose sur le numéro d'annonce de la sonde. Après le remplacement d'une sonde qui reprend l'identité de la précédente, ce numéro n'est croissant que si l'heure de la nouvelle sonde est fiable (chapitre 20, § 20.4.9) ; videz alors l'état de l'automatisme pour cette sonde ;
- **reprise** : le script vérifie que la sonde répond et a quitté l'état de maintenance ; il ne vérifie pas l'arrivée de nouveaux résultats de contrôle. Les alertes de contrôle, rétablies après la levée du silence, prennent le relais ;

- **validation** : comme tout exemple de cette documentation, adaptez ce script à votre chaîne de traitement des logs et validez-le dans un environnement de recette avant de l'utiliser en production.

18.15.5 Mise en sourdine dans Datadog, Dynatrace, Centreon et Nagios

Les règles communes du § 18.15.1 s'appliquent à chaque plateforme : une période récurrente par fenêtre de la sonde, dans son fuseau horaire, prolongée de la marge et limitée à la sonde concernée. Les sections suivantes indiquent le mécanisme propre à chaque plateforme.

Datadog

Datadog suspend les notifications de moniteurs au moyen de **temps d'arrêt** (*downtimes*), qui peuvent être récurrents.

1. Ouvrez la page **Manage Downtime**, puis **Schedule Downtime**.
2. Sélectionnez les moniteurs de la sonde **par tags de moniteur** (*monitor tags*), par exemple un tag que vous avez posé sur tous les moniteurs `sva.*`. Un temps d'arrêt par tags ne s'applique qu'aux moniteurs qui portent **tous** les tags sélectionnés.
3. Restreignez la **portée de groupe** (*group scope*) à la sonde concernée, par exemple `host:sonde-paris-01` pour des moniteurs groupés par `host` (§ 18.7.3). Sans portée de groupe, toutes les sondes sont suspendues.
4. Choisissez une planification **récurrente** : date et heure de début, **fuseau horaire** de la sonde, répétition hebdomadaire sur les jours de la fenêtre, et durée égale à celle de la fenêtre augmentée de la marge. Une règle de récurrence au format RRULE est également acceptée.
5. Enregistrez, puis vérifiez dans l'aperçu la liste des moniteurs concernés.

Par défaut, Datadog envoie une nouvelle notification si un moniteur est toujours en `ALERT`, `WARNING` ou `NO DATA` à l'expiration du temps d'arrêt : une sonde qui ne serait pas revenue après la fenêtre est donc signalée. Tout moniteur créé ou modifié ultérieurement et correspondant à la portée est automatiquement inclus. Les temps d'arrêt peuvent aussi être créés par l'API de Datadog, ce qui permet de les dériver de la configuration des sondes.

Dynatrace

Dynatrace suspend l'alerte au moyen de **fenêtres de maintenance** (*maintenance windows*).

1. Ouvrez **Settings** > **Maintenance windows** > **Monitoring, alerting, and availability**, puis **Create maintenance window**.
2. Nommez la fenêtre (par exemple « Activation sonde-paris-01 ») et choisissez le type **Planned**.
3. Choisissez une récurrence **hebdomadaire**, les jours et l'horaire de la fenêtre augmentés de la marge, et le **fuseau horaire** de la sonde.
4. Choisissez le comportement de détection : **Detect problems but don't alert** (les problèmes restent enregistrés, sans notification ; recommandé pour conserver une trace) ou **Disable problem detection** (aucun problème n'est détecté). L'option **Detect problems and alert** ne supprime rien et ne convient pas.
5. Limitez la **portée** à la sonde par un filtre (entité, tags d'entité ou zone de gestion). Sans filtre, la fenêtre s'applique à **tout l'environnement**.

> ! **Attention** — Les données de la sonde arrivent dans Dynatrace par OTLP (§ 18.8). Vérifiez dans votre environnement que les alertes définies sur les métriques `sva.*` (événements métriques, détecteurs d'anomalies) sont bien rattachées aux entités ou aux tags retenus dans la portée de la fenêtre. À défaut, la fenêtre n'aurait aucun effet sur ces alertes. Les libellés ci-dessus sont ceux de la documentation de Dynatrace en vigueur à la date de rédaction ; ils peuvent varier selon votre version.

Centreon

Centreon suspend les notifications au moyen de **temps d'arrêt récurrents** (*recurrent downtimes*).

1. Ouvrez **Monitoring** > **Downtimes** > **Recurrent Downtimes**, puis **Add**.
2. Renseignez le nom et la description, et cochez l'activation.
3. Ajoutez une **période** (*Add new period*) de type hebdomadaire (**Weekly**), avec les jours de la fenêtre, la plage horaire `HH:MM - HH:MM` (fenêtre augmentée de la marge) et le type **fixe** (*fixed*). Le type flexible ne démarre qu'à l'apparition d'un état non `OK` et ne couvre pas toute la fenêtre. Pour une fenêtre qui franchit minuit, déclarez deux périodes.
4. Dans les relations, liez le temps d'arrêt à l'**hôte** de la sonde (**Linked with Hosts**) **et** à ses **services** (**Linked with Services** ou **Linked with Service Groups**). Liez explicitement les services : la documentation de Centreon ne précise pas qu'un temps d'arrêt récurrent posé sur un hôte s'étend à ses services.
5. Enregistrez, puis vérifiez dans **Monitoring** > **Downtimes** que les temps d'arrêt apparaissent aux dates attendues.

Vérifiez que le fuseau horaire pris en compte par Centreon pour ces périodes correspond à celui de la sonde. Pendant le drainage, les services `check_sva` passent en `UNKNOWN` avec le texte `MAINTENANCE`, et la vérification de l'hôte fondée sur `/health` (§ 18.10.4) échoue, car `/health` répond alors avec le code HTTP 503 : ces deux effets sont couverts par le temps d'arrêt.

Nagios

Nagios Core suspend les notifications au moyen de **temps d'arrêt programmés** (*scheduled downtime*), créés depuis l'interface web (lien « Schedule downtime » de la fiche de l'hôte ou du service) ou par **commande externe**. Sa documentation ne décrit pas de temps d'arrêt récurrent natif : pour reproduire les fenêtres de la sonde, programmez chaque occurrence par une tâche planifiée (*cron*) qui écrit dans le fichier de commandes de Nagios.

Les deux commandes externes utiles sont :

Commande	Effet
`SCHEDULE_HOST_DOWNTIME;<hôte>;<début>;<fin>;<fixe>;<id déclencheur>;<durée>;<auteur>;<commentaire>`	Temps d'arrêt de l'hôte
`SCHEDULE_HOST_SVC_DOWNTIME;<hôte>;<début>;<fin>;<fixe>;<id déclencheur>;<durée>;<auteur>;<commentaire>`	Temps d'arrêt de tous les services de l'hôte

Les dates sont des horodatages Unix ; `<fixe>` vaut 1 pour un temps d'arrêt qui commence et finit exactement aux dates indiquées.

Exemple de script, exécuté par *cron* chaque mardi avant 04:00 pour la fenêtre fictive de `sonde-paris-01` (mardi à 04:00 pendant 1 heure, marge de 45 minutes). Le chemin du fichier de commandes est celui d'une installation de Nagios Core depuis les sources : adaptez-le à votre installation.

```shell
#!/bin/sh
# Temps d'arrêt de la sonde sonde-paris-01 pour sa fenêtre du jour (exemple fictif).
CMD=/usr/local/nagios/var/rw/nagios.cmd
HOTE=sonde-paris-01
MAINTENANT=$(date +%s)
# Début de la fenêtre dans le fuseau horaire de la sonde (GNU date)
DEBUT=$(TZ=Europe/Paris date -d 'today 04:00' +%s)
FIN=$((DEBUT + 6300))   # 1 h de fenêtre + 45 min de marge

printf '[%s] SCHEDULE_HOST_DOWNTIME;%s;%s;%s;1;0;0;supervision;Fenêtre de maintenance de la sonde\n'
    \
  "$MAINTENANT" "$HOTE" "$DEBUT" "$FIN" > "$CMD"
printf '[%s] SCHEDULE_HOST_SVC_DOWNTIME;%s;%s;%s;1;0;0;supervision;Fenêtre de maintenance de la
    sonde\n' \
  "$MAINTENANT" "$HOTE" "$DEBUT" "$FIN" > "$CMD"
```

Nagios envoie une notification de début (`DOWNTIMESTART`) et de fin (`DOWNTIMEEND`) de temps d'arrêt aux contacts qui les ont activées. Vérifiez le résultat dans la page **Downtime** de l'interface. Pour une activation hors fenêtre, programmez un temps d'arrêt ponctuel de la même manière, à partir des dates annoncées par la sonde.

18.15.6 Redondance : alertes de site

Une sonde unique ne supervise pas pendant son propre redémarrage. Pour les environnements critiques, le § 20.4.12, qui en décrit la mise en place, recommande au moins **deux sondes par site** exécutant les mêmes contrôles, avec des fenêtres de maintenance **qui ne se chevauchent pas**. Les alertes applicatives doivent alors porter sur **l'ensemble des sondes du site**, et non sur une sonde isolée. Deux alertes distinctes sont nécessaires, car un échec applicatif et une perte de couverture appellent des réactions différentes :

Alerte	Se déclenche lorsque	Signification
`ParcoursEnEchecSurLeSite`	Toutes les sondes **évaluables** du site voient le contrôle en `CRITICAL` , au moins une sonde étant évaluable	Incident probable de l'application ou du parcours supervisé
`CouvertureInsuffisanteSurLeSite`	Le nombre de sondes à résultat évaluable est inférieur à l'**effectif attendu** du site, y compris en l'absence totale de séries	La supervision du site repose sur moins de sondes que prévu, ou n'est plus assurée

Une sonde est **évaluable** pour un contrôle lorsqu'elle en fournit un résultat `OK` , `WARNING` ou `CRITICAL` et qu'elle n'est pas en maintenance. Les règles s'appuient sur la série `sva_check_status` (§ 18.3.3), qui vaut `0` pour `OK` , `1` pour `WARNING` , `2` pour `CRITICAL` et `3` pour `UNKNOWN` ; l'état `STALE` n'y est pas reporté : la série conserve le dernier état produit par une exécution. L'effectif attendu est un paramètre propre à chaque site, déclaré par une règle d'enregistrement `sva:site_effectif_attendu` , une par site : c'est le seul paramètre à adapter. Le groupe suivant, avec le label `site` ajouté par la configuration de collecte (§ 18.5.2), est identique au groupe `sondes-site` du fichier `exemples/prometheus/sva-regles-alerte.yml` :

```yaml
groups:
  - name: sondes-site
    rules:
      # PARAMÈTRE À ADAPTER : effectif attendu de chaque site, c'est-à-dire le nombre de
      # sondes du site qui exécutent les mêmes contrôles. Une règle par site (valeurs fictives).
      - record: sva:site_effectif_attendu
        expr: vector(2)
        labels:
          site: paris-dc1

      - record: sva:site_effectif_attendu
        expr: vector(2)
        labels:
          site: lyon-dc1

      # Toutes les sondes évaluables du site voient le contrôle en CRITICAL, au moins une
      # sonde étant évaluable. Le filtre <= 2 écarte UNKNOWN ; la clause unless écarte les
      # sondes en maintenance. Si aucune sonde n'est évaluable, l'expression ne renvoie rien :
      # cette situation relève de CouvertureInsuffisanteSurLeSite.
      - alert: ParcoursEnEchecSurLeSite
        expr: |
          min by (site, sva_check_id) (
            sva_check_status <= 2
              unless on (job, instance) sva_agent_maintenance == 1
          ) == 2
        for: 5m
        labels:
          severity: critical
        annotations:
```

```
      summary: "Contrôle {{ $labels.sva_check_id }} en CRITICAL sur toutes les sondes évaluables
du site {{ $labels.site }}"
      description: "Toutes les sondes du site {{ $labels.site }} qui fournissent un résultat
évaluable pour ce contrôle le voient en échec. Les sondes non évaluables (UNKNOWN, maintenance,
injoignables) sont signalées par CouvertureInsuffisanteSurLeSite."

  # Nombre de sondes à résultat évaluable inférieur à l'effectif attendu du site.
  # Première partie : par contrôle ; une sonde est comptée une seule fois par son label
  # instance, même si elle est collectée par plusieurs travaux (count by … instance) ; un
  # contrôle dont toutes les séries sont UNKNOWN ou en maintenance compte 0 sonde évaluable
  # (terme « or … * 0 »).
  # Seconde partie : absence totale de séries de contrôle pour un site attendu (toutes
  # les sondes du site injoignables) ; renvoie la valeur 0, sans limite de durée.
  # Une activation planifiée réduit la couverture pendant sa durée : le délai « for »
  # doit dépasser la durée totale mesurée d'une activation (documentation SYNOVEX, « Mises à jour
»),
  # 45 minutes ici à titre indicatif.
  # LIMITES : un contrôle qui disparaît de toutes les sondes du site, alors que d'autres
  # contrôles y restent présents, ne produit plus aucune série et n'est pas signalé ; une
  # série présente ne prouve pas une exécution récente (sva_check_status conserve le
  # dernier état, l'état STALE n'y est pas reporté). Le groupe facultatif
  # sondes-site-controles couvre ces deux cas pour les contrôles déclarés attendus.
  - alert: CouvertureInsuffisanteSurLeSite
    expr: |
      (
        (
            count by (site, sva_check_id) (
              count by (site, sva_check_id, instance) (
                sva_check_status <= 2
                  unless on (job, instance) sva_agent_maintenance == 1
              )
            )
          or
            count by (site, sva_check_id) (sva_check_status) * 0
        )
          < on (site) group_left ()
        sva:site_effectif_attendu
      )
      or on (site)
      (
        sva:site_effectif_attendu
          unless on (site)
        count by (site) (sva_check_status)
      ) * 0
    for: 45m
    labels:
      severity: warning
    annotations:
      summary: "Couverture insuffisante sur le site {{ $labels.site }}"
      description: "{{ $value }} sonde(s) du site {{ $labels.site }} fournissent un résultat
évaluable (contrôle : {{ $labels.sva_check_id }} ; vide si aucune série du site n'est reçue).
Vérifiez l'état des sondes du site et la valeur de sva:site_effectif_attendu."
```

Lecture des expressions :

- **`ParcoursEnEchecSurLeSite`** : le filtre `sva_check_status <= 2` écarte les séries `UNKNOWN`, qui ne disent rien de l'état de l'application. La clause `unless` écarte les sondes en maintenance : leur dernière valeur de `sva_agent_maintenance` est `1` ; pendant leur redémarrage, leurs séries cessent d'être actualisées, puis ne participent plus au calcul. Le minimum vaut `2` lorsque toutes les sondes restantes voient le contrôle en `CRITICAL`. Si aucune sonde n'est évaluable, l'expression ne renvoie rien : cette situation relève de l'alerte de couverture ;
- **`CouvertureInsuffisanteSurLeSite`** : le regroupement intermédiaire par `instance` compte chaque sonde une seule fois, même si elle est collectée par plusieurs travaux de collecte sous la même adresse. Le terme `or … * 0`

compte zéro sonde évaluable lorsque toutes les séries d'un contrôle sont `UNKNOWN` ou en maintenance. Le second terme, après `or on (site)`, signale avec la valeur `0` l'absence totale de séries pour un site attendu (toutes ses sondes injoignables), sans limite de durée.

Avec cette forme, l'activation d'une sonde n'interrompt pas la détection des pannes applicatives : l'autre sonde du site continue de les signaler. Les alertes propres à chaque sonde (`SondeInjoignable`, `SondeSilencieuse`) restent mises en sourdine pendant ses seules fenêtres.

Avec ces règles :

- un `CRITICAL` vu par une sonde alors que l'autre voit le contrôle en `OK` ne déclenche pas `ParcoursEnEchecSurLeSite` : il signale plutôt un problème local, suivi par les alertes de la sonde ;
- un `CRITICAL` sur une sonde et un `UNKNOWN` sur l'autre déclenchent les deux alertes : l'échec est observé par la seule sonde évaluable, et la couverture est réduite ;
- l'absence totale de séries pour un site attendu déclenche `CouvertureInsuffisanteSurLeSite` avec la valeur `0`, sans limite de durée ;
- une sonde collectée par deux travaux de collecte n'est comptée qu'une fois, à condition que son label `instance` soit le même dans les deux travaux. Collectée sous deux adresses différentes (nom et adresse IP, par exemple), elle serait comptée deux fois : collectez chaque sonde sous une seule adresse ;
- une activation planifiée réduit la couverture pendant sa durée : le délai `for` de l'alerte de couverture (45 minutes, indicatif) doit dépasser la durée totale mesurée d'une activation (chapitre 20, § 20.4.5) ; à défaut, neutralisez cette seule alerte pendant les fenêtres des sondes du site (§ 18.15.1). Ne mettez pas `ParcoursEnEchecSurLeSite` en sourdine pendant les fenêtres : c'est elle qui détecte un incident pendant la maintenance d'une sonde.

> ! **Attention** — Ces deux règles raisonnent sur les séries présentes. Elles ne signalent pas un contrôle qui disparaît de **toutes** les sondes du site (contrôle supprimé ou désactivé, configuration divergente) tant que d'autres contrôles du site restent présents : aucune série ne le représente plus. Elles ne vérifient pas non plus l'âge du résultat : `sva_check_status` conserve le dernier état d'un contrôle, et l'état `STALE` n'y est pas reporté. Une sonde dont un contrôle ne s'exécute plus reste donc comptée comme évaluable avec son dernier résultat. Le catalogue des métriques ne comporte pas d'horodatage de dernière exécution. L'extension facultative ci-dessous couvre ces deux cas pour les contrôles que vous déclarez attendus.

Extension facultative : couverture de chaque contrôle attendu. Le groupe `sondes-site-controles` du même fichier déclare l'inventaire des contrôles attendus sur chaque site, par une règle d'enregistrement `sva:site_controle_attendu` par couple site/contrôle, sur le modèle de `sva:site_effectif_attendu`. Son alerte `ControleAttenduNonCouvertSurLeSite` compte, pour chaque contrôle de l'inventaire, les sondes **fraîchement évaluables** : évaluables au sens précédent, et ayant terminé au moins une exécution de ce contrôle pendant une fenêtre de fraîcheur, mesurée par le compteur `sva_check_runs_total`. Elle se déclenche lorsque ce nombre est inférieur à l'effectif attendu du site, y compris lorsque le contrôle a disparu de toutes les sondes (valeur `0`). Le groupe suivant est identique à celui du fichier :

```yaml
groups:
  - name: sondes-site-controles
    rules:
      # PARAMÈTRE À ADAPTER : inventaire des contrôles attendus sur chaque site. Une règle
      # par couple site/contrôle (valeurs fictives). Un contrôle absent de cet inventaire
      # n'est pas surveillé par ce groupe.
      - record: sva:site_controle_attendu
        expr: vector(1)
        labels:
          site: paris-dc1
          sva_check_id: boutique-connexion

      - record: sva:site_controle_attendu
```

```
        expr: vector(1)
        labels:
          site: lyon-dc1
          sva_check_id: boutique-connexion

      # Nombre de sondes fraîchement évaluables inférieur à l'effectif attendu, pour un
      # contrôle de l'inventaire, y compris lorsque ce contrôle a disparu de toutes les
      # sondes du site (valeur 0). Chaque sonde est comptée une seule fois (label instance).
      # PARAMÈTRE À ADAPTER : fenêtre de fraîcheur [15m], au moins deux intervalles de
      # planification du contrôle le plus espacé, comme l'état STALE (documentation SYNOVEX,
      # « Les états »), et au
      # moins deux intervalles de collecte. Pour des contrôles d'intervalles très différents,
      # dupliquez la règle avec une fenêtre propre et un filtre sur sva_check_id.
      # Le délai « for » doit dépasser la durée totale mesurée d'une activation augmentée de
      # la fenêtre de fraîcheur (45 minutes ici, à titre indicatif).
      - alert: ControleAttenduNonCouvertSurLeSite
        expr: |
          (
              count by (site, sva_check_id) (
                count by (site, sva_check_id, instance) (
                  (
                    sva_check_status <= 2
                      unless on (job, instance) sva_agent_maintenance == 1
                  )
                    and on (job, instance, sva_check_id)
                  (
                    sum by (job, instance, sva_check_id) (increase(sva_check_runs_total[15m])) > 0
                  )
                )
              )
            or
              sva:site_controle_attendu * 0
          )
            < on (site) group_left ()
          sva:site_effectif_attendu
            and on (site, sva_check_id)
          sva:site_controle_attendu
        for: 45m
        labels:
          severity: warning
        annotations:
          summary: "Contrôle attendu {{ $labels.sva_check_id }} insuffisamment couvert sur le site {{
    $labels.site }}"
          description: "{{ $value }} sonde(s) du site {{ $labels.site }} ont fourni récemment un
    résultat évaluable pour le contrôle attendu {{ $labels.sva_check_id }}. Vérifiez que le contrôle
    existe et s'exécute sur chaque sonde du site, et la valeur de sva:site_effectif_attendu."
```

Limites de cette extension : l'inventaire est tenu à la main et doit suivre chaque ajout ou suppression de contrôle ; une seule fenêtre de fraîcheur s'applique à tous les contrôles de la règle, ce qui impose de dupliquer la règle pour des contrôles d'intervalles très différents ; l'identité d'une sonde repose sur son label `instance`, avec la réserve indiquée plus haut pour une sonde collectée sous deux adresses ; enfin, `ParcoursEnEchecSurLeSite` n'applique pas cette condition de fraîcheur : un dernier résultat `CRITICAL` ancien peut la maintenir active ; pour un contrôle de l'inventaire, l'alerte de l'extension signale alors que ce résultat n'est plus récent. Si vous n'utilisez pas l'extension, supprimez le groupe `sondes-site-controles` du fichier : ses valeurs fictives déclencheraient sinon des alertes.

En mode envoi, remplacez le label `site` par le label issu de l'attribut de ressource `sva.site` (`sva_site`), après sa promotion en label (§ 18.5.3). Ces règles ont été vérifiées avec `promtool test rules` sur des séries simulées : sondes `OK`, `CRITICAL` sur toutes les sondes, `CRITICAL` et `UNKNOWN`, `UNKNOWN` seul, sonde en maintenance et absence totale de séries ; pour l'extension et le comptage par sonde, contrôle attendu disparu seul, sonde collectée par deux travaux, jauge périmée (exécutions arrêtées) et maintenance suivie d'une reprise.

18.16 Dépannage des exports

Le tableau suivant regroupe les incidents d'export les plus fréquents. Commencez toujours par le test intégré de la destination (§ 18.4.4, étape 3) : il renvoie le code HTTP obtenu, le message d'erreur de la plateforme et la latence. Le chapitre 19 (§ 19.8) complète ce tableau dans la démarche générale de diagnostic de la sonde.

Symptôme	Cause probable	Vérification	Action
HTTP 401	Clé ou jeton d'ingestion absent, invalide ou révoqué ; format d'en-tête erroné	Test de la destination ; secret référencé dans `headers`	Remplacer le secret ; vérifier le format de l'en-tête : `Api-Token` (Dynatrace, jeton classique), `dd-api-key` (Datadog), `Basic` (Grafana Cloud, § 18.6.2)
HTTP 403	Jeton sans les étendues d'ingestion requises (§ 18.8.1) ; pour Datadog, adresse qui ne correspond pas au site du compte (§ 18.7.2)	Étendues du jeton ; site Datadog	Ajouter les étendues nécessaires ; utiliser l'adresse du bon site
HTTP 400 sur les métriques vers Dynatrace ou l'intake direct de Datadog	Temporalité cumulative envoyée à une plateforme qui n'accepte que le delta (§ 18.1.5)	Paramètre `metrics.temporality`	Utiliser le préréglage `dynatrace` ou `datadog-agentless` (temporalité `delta`)
Succès partiel, certaines métriques absentes dans Dynatrace	Requête mêlant données acceptées et refusées (type non pris en charge, temporalité)	Journaux d'export de la sonde ; `sva.export.failures`	Utiliser le préréglage `dynatrace` (delta, histogrammes explicites)
HTTP 404	Chemin d'adresse incorrect, notamment pour le récepteur OTLP de Prometheus	Adresse complète testée	Pour Prometheus, déclarer l'adresse complète `/api/v1/otlp/v1/metrics` dans `signalEndpoints.metrics` (§ 18.5.3)
HTTP 413	Lot trop volumineux	Volume des envois	Désactiver `metrics.stepHistograms`, allonger `metrics.exportInterval` ; si l'erreur persiste, contacter le support
HTTP 429	Limitation de débit ou quota d'ingestion atteint	En-tête `Retry-After` ; consommation dans la plateforme	La sonde réessaie d'elle-même ; réduire le volume ou ajuster le quota
HTTP 5xx, erreurs réseau	Plateforme indisponible ; proxy ou pare-feu	Test de la destination ; flux sortant vers l'adresse	Ouvrir le flux sortant (annexe E) ; la sonde réessaie pendant `retry.maxElapsed`, puis abandonne le lot (`sva.export.dropped`, § 18.12)
Cible Prometheus à l'état `DOWN` (mode collecte)	Pare-feu ; jeton absent, expiré ou de rôle insuffisant (`metrics` au minimum) ; certificat de la sonde non reconnu	Page des cibles de Prometheus ; `curl` vers `/metrics` (§ 18.5.2, étape 3)	Ouvrir le port 443 depuis Prometheus ; utiliser un jeton `metrics` ; renseigner `tls_config.ca_file`
Données envoyées au récepteur OTLP de Prometheus, mais invisibles	Récepteur non activé ; noms traduits avec suffixes ; labels de site absents faute de promotion	Option `--web.enable-otlp-receiver` ; noms des séries (§ 18.3.3)	Activer le récepteur ; rechercher les noms traduits ; promouvoir `sva.site` (§ 18.5.3)

Symptôme	Cause probable	Vérification	Action
Métriques présentes, mais aucun tableau de bord ni aucune alerte	Comportement normal : l'ingestion ne crée ni tableau de bord ni alerte (§ 18.2)	—	Importer les modèles fournis (`exemples/grafana/`, `exemples/prometheus/`) ou créer vos alertes
`check_sva` renvoie `UNKNOWN`	Sonde injoignable ; jeton invalide ; identifiant de contrôle erroné ; texte `MAINTENANCE` : activation d'une mise à jour en cours (§ 18.15.1)	Exécuter le plugin en ligne de commande sur le serveur Nagios	Corriger le jeton (rôle `metrics`), l'identifiant (`--check`) ou le flux HTTPS
Trous dans les courbes après une coupure	Comportement documenté : les données ne sont pas rejouées après une indisponibilité de la destination (§ 18.12)	`sva.export.dropped`	Consulter l'historique local de la sonde ; pour réduire les pertes, utiliser un Collector à file persistante (§ 18.9)
Absence de données de la sonde dans la plateforme alors qu'elle fonctionne	Export interrompu (lignes précédentes) ; heure de la sonde décalée ; activation d'une mise à jour en cours	Test de la destination ; synchronisation horaire de la sonde ; événements `maintenance.started` et `maintenance.ended`	Corriger l'export ou la synchronisation NTP ; pendant une activation, l'absence de données est normale (§ 18.15.1)
Alertes reçues pendant la maintenance de la sonde	Périodes de maintenance absentes de la plateforme, ou non alignées sur les fenêtres de la sonde	Heure des alertes comparée aux fenêtres de la sonde	Aligner les périodes de maintenance de la plateforme (§ 18.15)

Chapitre 19 — Diagnostic et résolution des problèmes

Ce chapitre aide à identifier et corriger les incidents les plus courants. Il est organisé par domaine : matériel, mise en service, réseau, contrôles, scénarios navigateur, exports, stockage et mises à jour. Chaque domaine présente un tableau qui associe un **symptôme** à ses **causes possibles**, aux **vérifications** à mener et aux **actions correctives**.

Le chapitre ne remplace pas les chapitres de configuration. Il y renvoie lorsqu'une correction suppose de modifier un paramètre.

> **Information** — Les exemples de ce chapitre utilisent des noms et adresses fictifs (`sonde-paris-01`, `192.0.2.50`, `example.com`). Remplacez-les par les valeurs de votre environnement.

19.1 Méthode générale de diagnostic

19.1.1 Principe

Un incident de supervision peut avoir trois origines distinctes, qu'il faut séparer avant toute action :

1. **La ressource supervisée** : l'application, l'API, la base ou le service surveillé est réellement défaillant. C'est le cas nominal : la sonde remplit son rôle en le signalant.
2. **Le chemin entre la sonde et la ressource** : réseau, résolution DNS, pare-feu, proxy, certificat. La ressource fonctionne peut-être pour ses utilisateurs, mais pas depuis l'emplacement de la sonde.
3. **La sonde elle-même** : alimentation, stockage, mémoire, configuration du contrôle, script de scénario devenu obsolète.

L'état `UNKNOWN` est réservé aux situations où la sonde n'a pas pu évaluer le contrôle (troisième cas). Un état `CRITICAL` signale en principe un problème de la ressource ou du chemin réseau (premier et deuxième cas). Cette distinction, décrite au chapitre 8, est le premier indice à exploiter.

19.1.2 Démarche recommandée

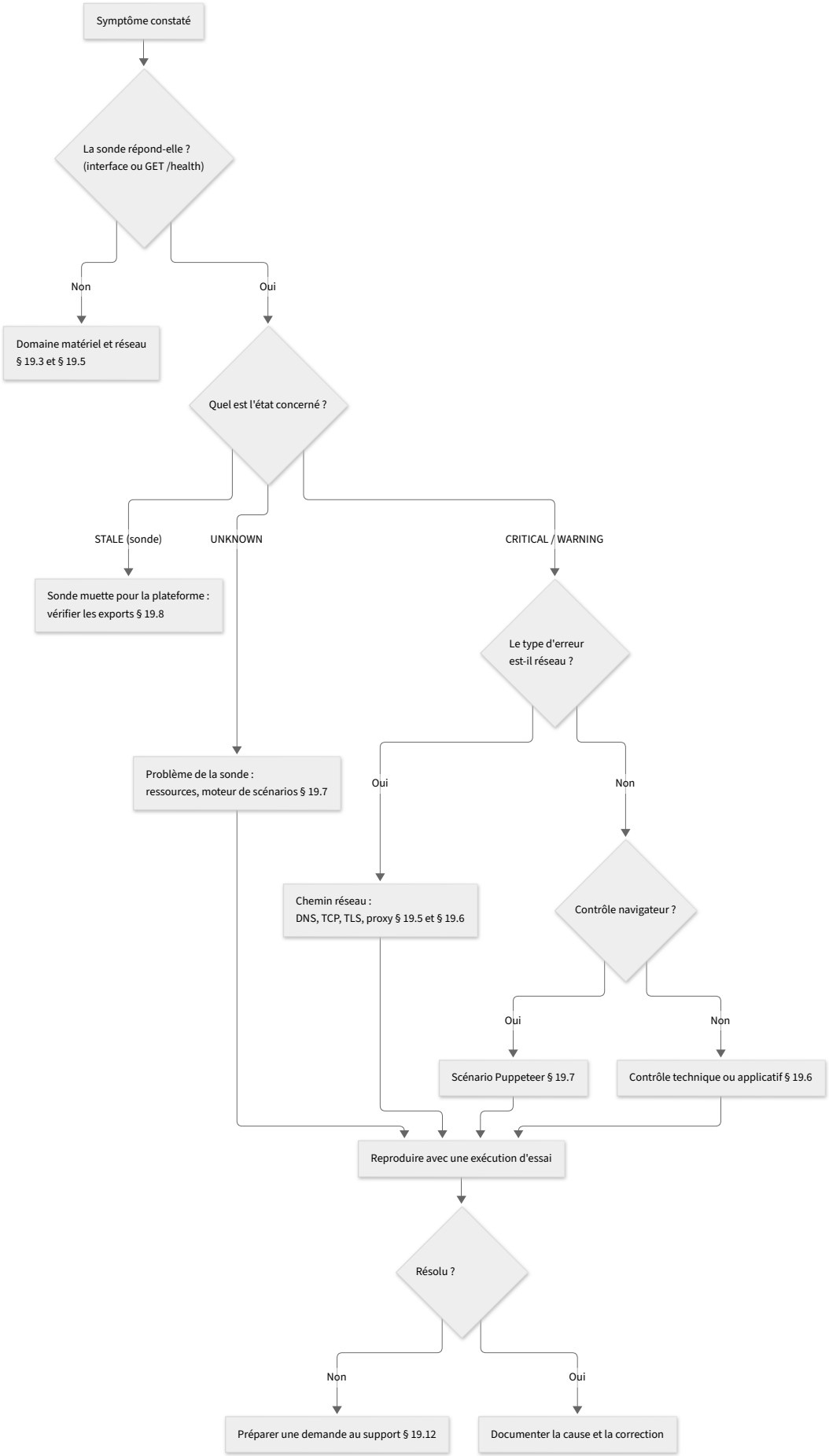


Schéma 19-1 — Démarche de diagnostic

En pratique :

1. **Qualifier le symptôme** : quel contrôle, depuis quand, sur quelle sonde, avec quel état et quel type d'erreur (`error.type`).
2. **Consulter le détail de la dernière exécution en échec** : étape concernée, message, capture d'écran et journal de console pour un scénario navigateur (voir chapitre 17).
3. **Comparer avec l'historique** : un échec isolé, un bagotage ou une rupture nette n'appellent pas la même analyse.
4. **Vérifier si plusieurs contrôles sont touchés en même temps**. Si tous les contrôles d'une même sonde échouent simultanément, la cause se trouve probablement côté sonde ou réseau local, et non dans les applications supervisées.
5. **Reproduire** avec une exécution d'essai (`POST /api/v1/checks/{id}/runs?dryRun=true`). Elle n'enregistre pas d'état et n'exporte rien, ce qui évite de fausser l'historique pendant le diagnostic.
6. **Corriger, puis vérifier** par une nouvelle exécution d'essai avant de réactiver le contrôle ou d'attendre la prochaine échéance.

> ✓ **Conseil** — Notez systématiquement l'heure UTC de l'exécution en échec et son identifiant (`runId`). Ces deux informations permettent de retrouver l'événement dans les journaux, les traces OpenTelemetry et les plateformes d'observabilité.

19.2 Outils de diagnostic disponibles

19.2.1 Interface d'administration

L'interface d'administration (chapitre 9) est l'outil de premier niveau. Elle permet de :

- consulter l'état de chaque contrôle et l'historique des exécutions ;
- ouvrir le détail d'une exécution : étapes, durée de chaque étape, type d'erreur, message, artefacts (capture d'écran, journal de console) ;
- déclencher une exécution manuelle ou d'essai ;
- consulter l'état de santé de la sonde : ressources, modules, destinations d'export ;
- consulter le journal d'audit (rôle `admin`).

19.2.2 API REST

L'API offre les mêmes informations sous une forme exploitable par script. Les appels les plus utiles en diagnostic sont les suivants.

Appel	Rôle minimal	Utilité en diagnostic
`GET /health`	aucun	Vérifie que le moteur principal répond. Ne donne aucune information sensible
`GET /api/v1/system`	`viewer`	Version, édition, ressources, état des modules et des destinations d'export
`GET /api/v1/checks/{id}/status`	`metrics`	Dernier état consolidé d'un contrôle
`GET /api/v1/runs?checkId=…&status=CRITICAL`	`viewer`	Historique des échecs d'un contrôle
`GET /api/v1/runs/{runId}`	`viewer`	Détail complet d'une exécution

Appel	Rôle minimal	Utilité en diagnostic
`GET /api/v1/runs/{runId}/artifacts/{name}`	`viewer`	Téléchargement d'une capture ou d'un journal de console
`POST /api/v1/checks/{id}/runs?dryRun=true`	`operator`	Exécution d'essai
`POST /api/v1/exporters/{id}/test`	`admin`	Test d'une destination d'export avec diagnostic détaillé
`GET /api/v1/audit-events`	`admin`	Modifications récentes de configuration

Exemple : vérifier que la sonde répond, puis afficher son état de santé. Dans cet exemple fictif, le jeton d'API n'apparaît pas dans la ligne de commande : il est lu dans le fichier d'en-tête protégé `$HOME/.sva/entete-auth`, créé une fois selon le chapitre 9 (§ 9.3.3). Le certificat de la sonde, récupéré puis vérifié selon le chapitre 6 (§ 6.4.4), est enregistré dans `sonde-ca.pem`.

```bash
# Vivacité du moteur principal (aucune authentification requise)
curl --cacert sonde-ca.pem https://sonde-paris-01.example.internal/health

# État de santé détaillé (rôle viewer)
curl --cacert sonde-ca.pem \
     -H @"$HOME/.sva/entete-auth" \
     https://sonde-paris-01.example.internal/api/v1/system
```

Exemple : lister les dernières exécutions en échec d'un contrôle.

```bash
curl --cacert sonde-ca.pem \
     -H @"$HOME/.sva/entete-auth" \
     "https://sonde-paris-01.example.internal/api/v1/runs?checkId=boutique-
    connexion&status=CRITICAL&limit=20"
```

> **Attention** — N'utilisez pas l'option `curl -k` (`--insecure`, vérification du certificat désactivée) pour vos appels courants : elle supprime la protection contre l'interception, y compris celle de votre jeton d'API. Utilisez l'option `--cacert`, avec le certificat de la sonde ou celui de l'autorité de certification de votre entreprise (chapitres 6 et 7).

19.2.3 Journaux

La sonde produit trois catégories de traces :

Source	Contenu	Accès
Journal de chaque exécution	Étapes, messages `log.info` des scénarios, sortie des plugins Nagios, erreurs (secrets masqués)	Interface, `GET /api/v1/runs/{runId}`
Journal d'audit	Opérations d'administration : connexions, modifications, téléversements, mises à jour	Interface (`admin`), `GET /api/v1/audit-events`
Journaux système	Messages des services de la sonde et du système d'exploitation	SSH (si activé), export OTLP des logs vers votre plateforme

> **Information** — Par défaut, les journaux système sont conservés **en mémoire** pour limiter l'usure de la microSD (voir chapitre 20). Ils sont donc perdus à chaque redémarrage. Pour analyser un incident ayant entraîné un redémarrage, exportez les logs vers une plateforme d'observabilité (chapitre 18) ou activez temporairement la conservation persistante avec le paramètre `logging.persistentUntil`, qui fixe l'échéance de la persistance. La sonde revient automatiquement aux journaux en mémoire à cette échéance, et au plus tard 72 heures après l'activation.

19.2.4 Accès SSH (diagnostic avancé)

L'accès SSH est **désactivé par défaut**. Seul un administrateur (rôle `admin`) de la sonde peut l'activer, depuis l'interface d'administration ou dans la section `ssh` des paramètres, et toujours pour une durée limitée : l'échéance (`enabledUntil`) est obligatoire, de 4 heures par défaut et de 72 heures au maximum. À l'échéance, SSH se désactive automatiquement et les sessions ouvertes sont fermées. L'activation, les connexions et la désactivation sont enregistrées dans le journal d'audit (voir chapitre 7, § 7.9.4). Il est destiné aux administrateurs système expérimentés et au support de l'éditeur. Si SSH a été verrouillé (`ssh.locked`), il ne peut plus être activé avant une réinitialisation usine.

> **Avertissement** — Une session SSH donne un accès étendu au système de la sonde. Ne modifiez pas les fichiers du produit et n'installez pas de logiciels supplémentaires : ces opérations ne sont pas prises en charge et peuvent être annulées par une mise à jour, le système étant installé en lecture seule. Désactivez SSH à la fin de l'intervention, sans attendre l'échéance automatique.

Les commandes ci-dessous sont des commandes Linux standard, disponibles sur Raspberry Pi OS. Elles ne modifient rien.

Commande	Objet
`systemctl status sva-agent sva-runner`	État des services du moteur principal et du moteur de scénarios
`journalctl -u sva-agent --since "1 hour ago"`	Journal du moteur principal sur la dernière heure
`journalctl -u sva-runner -n 200`	200 dernières lignes du journal du moteur de scénarios
`journalctl -k -b`	Messages du noyau depuis le dernier démarrage
`dmesg --level=err,warn`	Erreurs et avertissements du noyau (carte microSD, mémoire, USB)
`ip addr show eth0`	Adresse IP de l'interface Ethernet
`ip route`	Passerelle par défaut
`getent hosts www.example.com`	Résolution d'un nom selon la configuration du système
`dig www.example.com`	Interrogation DNS détaillée (outil présent dans l'image de la sonde)
`ping -c 4 192.0.2.1`	Joignabilité ICMP
`curl -v https://www.example.com/`	Requête HTTPS détaillée (DNS, TLS, réponse)
`timedatectl`	Heure système et état de la synchronisation NTP
`df -h`	Espace disque par partition
`free -h`	Mémoire disponible
`uptime`	Durée de fonctionnement et charge
`vcgencmd get_throttled`	Indicateurs de sous-tension et de bridage thermique (Raspberry Pi)

Commande	Objet
`vcgencmd measure_temp`	Température du processeur (Raspberry Pi)

> **Information** — Les services de la sonde s'appellent `sva-agent` (moteur principal) et `sva-runner` (moteur de scénarios et environnement isolé). Ces noms techniques sont stables. Le chemin du répertoire de données de la sonde est indiqué dans les notes de version.

Interpréter `vcgencmd get_throttled`

La commande renvoie une valeur hexadécimale, par exemple `throttled=0x50005`. Chaque bit a une signification documentée par Raspberry Pi :

Bit	Valeur	Signification
0	`0x1`	Sous-tension **actuellement** détectée
1	`0x2`	Fréquence du processeur actuellement plafonnée
2	`0x4`	Processeur actuellement bridé
3	`0x8`	Limite thermique logicielle actuellement atteinte
16	`0x10000`	Sous-tension survenue depuis le démarrage
17	`0x20000`	Plafonnement de fréquence survenu depuis le démarrage
18	`0x40000`	Bridage survenu depuis le démarrage
19	`0x80000`	Limite thermique logicielle atteinte depuis le démarrage

La valeur `0x0` indique qu'aucun problème n'a été détecté depuis le démarrage. Toute autre valeur appelle une vérification de l'alimentation ou du refroidissement (§ 19.3).

19.2.5 Métriques de supervision interne

La sonde publie des métriques sur sa propre santé (voir chapitre 18). Elles sont souvent l'indice le plus rapide :

Métrique	Ce qu'elle révèle
`sva.agent.up`	Absence de signal : sonde éteinte, injoignable ou export interrompu
`sva.agent.scheduler.lag`	Retard de la planification : sonde saturée
`sva.agent.queue.depth`	Contrôles en attente d'exécution
`sva.browser.launches` (issue en échec)	Problèmes de lancement de Chromium
`sva.export.failures`, `sva.export.dropped`	Destinations d'export en erreur, données perdues
`sva.storage.usage`	Remplissage de la base et des artefacts
`sva.dot1x.certificate.expiry`	Échéance du certificat client 802.1X ; un avertissement est émis 30 jours avant son expiration
`sva.hw.temperature`, `sva.hw.throttled`	Surchauffe, sous-tension

19.3 Démarrage et matériel

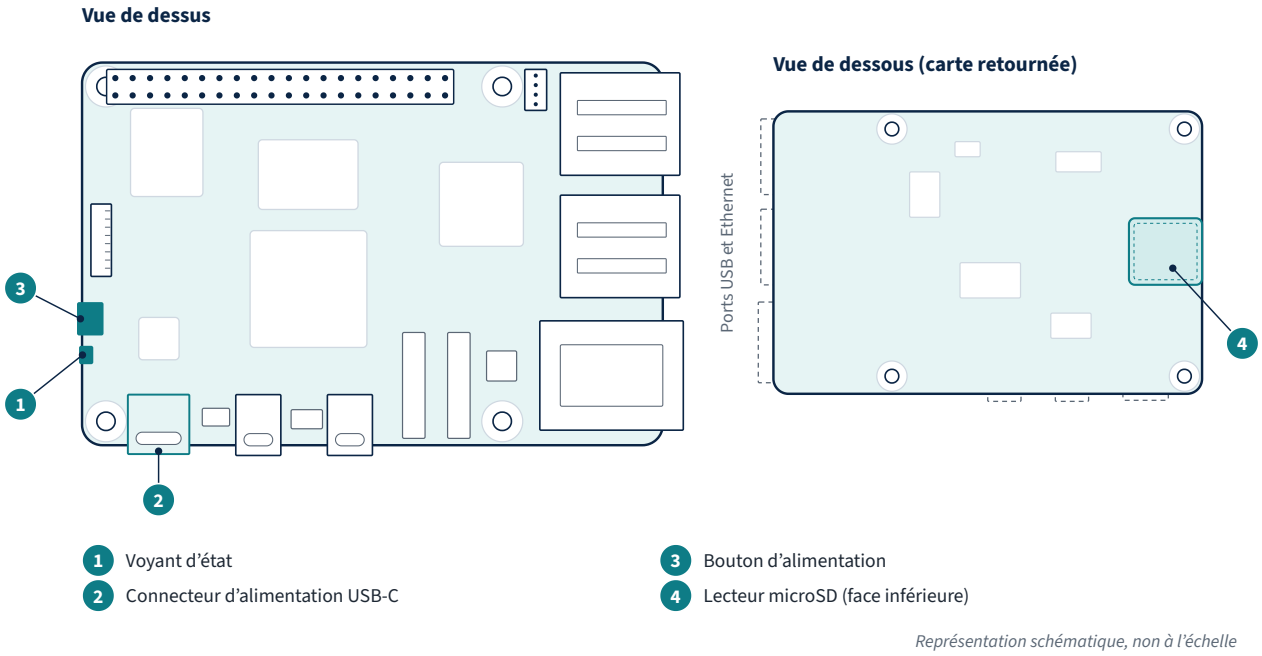


Figure 19-1 — Raspberry Pi 5 : emplacement du voyant d'état, du connecteur d'alimentation USB-C, du bouton d'alimentation et du lecteur microSD

Symptôme	Causes possibles	Vérifications	Actions correctives
Aucun voyant allumé à la mise sous tension	Alimentation absente ou défectueuse ; câble USB-C endommagé ; prise électrique sans courant. L'absence de voyant n'implique pas forcément l'absence d'alimentation : le comportement du voyant dépend aussi du micrologiciel, de la configuration et de l'état de démarrage	Tester la prise avec un autre appareil ; vérifier que le connecteur USB-C est bien enfoncé ; essayer une autre alimentation conforme ; brancher un écran HDMI pour vérifier si la console de démarrage s'affiche ; interroger `/health` si la sonde est déjà connue sur le réseau	Utiliser l'alimentation recommandée (27 W USB-C, 5,1 V / 5 A, chapitre 4) ; remplacer le câble
Le voyant s'allume mais la sonde n'apparaît jamais sur le réseau	Carte microSD absente, mal insérée ou illisible ; premier démarrage encore en cours ; câble Ethernet débranché	Vérifier l'insertion de la carte (chapitre 5) ; patienter le temps du premier démarrage, de l'ordre de quelques minutes à titre indicatif ; vérifier les voyants du port Ethernet et du commutateur	Réinsérer la carte hors tension ; brancher le câble avant la mise sous tension ; voir § 19.4 et § 19.5
Carte microSD non reconnue (la sonde ne démarre pas, comportement de voyant indiquant une absence de système)	Carte mal insérée, contacts sales, carte défaillante, image corrompue	Retirer la carte hors tension et l'inspecter ; la lire sur un poste de travail : la partition de démarrage, étiquetée `SVA-BOOT`, doit apparaître	Nettoyer délicatement les contacts ; réinsérer ; si la partition n'est pas lisible, remplacer la carte (chapitre 20, § 20.6)

Symptôme	Causes possibles	Vérifications	Actions correctives
Clignotements répétitifs du voyant d'état	Code de diagnostic du chargeur d'amorçage du Raspberry Pi (par exemple, image système introuvable)	Compter la séquence de clignotements longs et courts et la comparer aux codes publiés par Raspberry Pi	Selon le code (tableau ci-dessous) : réinsérer ou remplacer la carte, vérifier l'alimentation. L'image de la sonde n'ajoute aucune signalisation propre : seuls les codes standard du Raspberry Pi s'appliquent
La sonde redémarre spontanément	Sous-tension ; surchauffe ; watchdog déclenché par un blocage ; alimentation instable (multiprise surchargée)	`vcgencmd get_throttled` (bit 16) ; `journalctl -k -b -1` si les journaux sont persistants ; métrique `sva.hw.throttled`	Remplacer l'alimentation ; brancher la sonde sur un circuit protégé, idéalement un onduleur ; voir surchauffe ci-dessous
Sous-tension signalée (`sva.hw.throttled` avec `sva.hw.reason` = `undervoltage`, bit `0x1` ou `0x10000`)	Alimentation non conforme ; câble trop long ou de section insuffisante ; périphériques USB gourmands	`vcgencmd get_throttled`	Utiliser l'alimentation officielle 27 W et le câble d'origine ; débrancher les périphériques USB inutiles
Surchauffe ou bridage thermique (`sva.hw.temperature` élevée, bit `0x8` ou `0x80000`)	Refroidissement absent ou ventilateur bloqué ; boîtier confiné ; local trop chaud ; parcours navigateur très nombreux	`vcgencmd measure_temp` ; inspection visuelle du ventilateur ; température du local	Installer le refroidisseur actif officiel ou un boîtier ventilé (obligatoire pour l'édition Enterprise, recommandé pour les autres, chapitre 4) ; dégager les aérations ; déplacer la sonde ; réduire la concurrence navigateur (chapitre 12)
Performances dégradées sans alerte matérielle	Mémoire saturée ; carte microSD lente ou en fin de vie ; trop de contrôles pour l'édition et la carte	`free -h` ; `sva.agent.scheduler.lag` ; `dmesg` (erreurs d'entrée/sortie `mmc`)	Réduire la fréquence des contrôles ; répartir sur une seconde sonde ; demander une carte de remplacement (§ 20.6.4) ou ajouter un SSD NVMe (§ 20.3.4), recommandé surtout pour l'édition Enterprise
Erreurs d'entrée/sortie dans `dmesg` (`mmc0: … error`, `I/O error`, `EXT4-fs error`)	Carte microSD en fin de vie ou corrompue après coupure ; carte autre que celle fournie par l'éditeur	`dmesg --level=err`	Sauvegarder immédiatement la configuration (chapitre 20, § 20.5) ; demander une carte de remplacement (§ 20.6.4)

Information — Les couleurs et clignotements du voyant d'état sont des **indices matériels** produits par le Raspberry Pi et son chargeur d'amorçage. Ils ne renseignent pas sur l'état de l'application : une activité du voyant liée aux accès au stockage ne prouve pas que le moteur de supervision fonctionne. Recoupez toujours ces indices avec la console HDMI (messages de démarrage, adresse et empreinte affichées) et avec la réponse de `GET /health` (§ 19.2.2). L'image de la sonde n'ajoute aucune signalisation propre au voyant.

Le tableau suivant reprend les codes de clignotement standard publiés par Raspberry Pi dans sa documentation officielle (section *LED warning flash codes*). Un code se lit comme un nombre de clignotements longs suivi d'un nombre de clignotements courts. En cas de doute, la documentation officielle de Raspberry Pi fait foi.

Longs	Courts	Signification
0	3	Échec générique du démarrage
0	4	Fichier `start*.elf` introuvable
0	7	Image du noyau introuvable
0	8	Défaillance de la mémoire vive (SDRAM)
0	9	Mémoire vive (SDRAM) insuffisante
0	10	Système à l'arrêt (état HALT)
1	2	Surintensité détectée sur la carte microSD
2	1	Partition qui n'est pas au format FAT
2	2	Échec de lecture de la partition
2	3	Partition étendue qui n'est pas au format FAT
2	4	Signature ou empreinte de fichier non conforme
3	1	Erreur de l'EEPROM SPI
3	2	EEPROM SPI protégée en écriture
3	3	Erreur I2C
3	4	Configuration de démarrage sécurisé invalide
4	3	Composant RP1 introuvable
4	4	Type de carte non pris en charge
4	5	Erreur fatale du micrologiciel
4	6	Défaillance d'alimentation de type A
4	7	Défaillance d'alimentation de type B

> **Avertissement** — Ne retirez jamais la carte microSD et ne débranchez jamais l'alimentation pendant le fonctionnement de la sonde. Arrêtez-la proprement d'abord (chapitre 20, § 20.2). Un retrait à chaud peut corrompre la carte.

19.4 Mise en service

Symptôme	Causes possibles	Vérifications	Actions correctives
Impossible de trouver l'adresse IP de la sonde	DHCP inactif sur le réseau ; port de commutateur désactivé ou dans un autre VLAN ; nom mDNS non résolu par le poste	Rechercher l'adresse MAC dans la table de baux DHCP ; tester `sva-XXXXXX.local` depuis un poste du même réseau (l'annonce mDNS est active par défaut, sauf si elle a été désactivée dans la section `network`) ; brancher un écran HDMI, qui affiche l'adresse IP et le nom mDNS	Réserver une adresse DHCP pour l'adresse MAC ; ou provisionner une adresse statique via `provision.yaml` (chapitre 6)
Le navigateur affiche un avertissement de sécurité à la connexion	Comportement normal : le certificat de la sonde est autosigné	Comparer l'empreinte SHA-256 affichée par le navigateur avec celle du fichier `setup-code.txt` ou de l'écran HDMI (chapitre 6, § 6.4.4)	Accepter le certificat après vérification de l'empreinte ; ensuite, installer un certificat émis par l'autorité de certification de l'entreprise (chapitre 7)
L'empreinte du certificat ne correspond pas	Connexion à un autre équipement (erreur d'adresse) ; interception par un proxy d'inspection TLS ; tentative d'usurpation	Vérifier l'adresse saisie ; contourner le proxy du poste pour l'adresse de la sonde	Ne pas poursuivre tant que l'empreinte ne correspond pas ; exclure l'adresse de la sonde de l'inspection TLS
Code d'installation refusé	Faute de frappe (confusion entre caractères proches) ; code d'une autre sonde (encart ou étiquette d'un autre produit) ; assistant déjà terminé ; blocage temporaire après 5 saisies erronées : 15 minutes, durée doublée à chaque nouvelle série d'échecs	Relire `setup-code.txt` sur la partition `SVA-BOOT` de la microSD ou l'écran HDMI, qui affiche le code tant que la sonde n'est pas initialisée ; en cas de blocage, attendre la fin du délai avant de saisir de nouveau le code ; vérifier si un compte administrateur existe déjà	Utiliser le code actuel ; si l'assistant a déjà été terminé, se connecter avec le compte administrateur créé ; en cas de perte, voir réinitialisation (chapitre 20, § 20.10)
L'assistant ne s'affiche pas, la page de connexion apparaît directement	La sonde a été provisionnée par `provision.yaml` ou l'assistant a déjà été terminé	Lire `provision-result.txt` sur la partition `SVA-BOOT` de la microSD	Se connecter avec le compte défini dans le fichier de provisionnement
Le provisionnement n'a pas été appliqué	Fichier mal nommé (extension masquée par l'explorateur, par exemple `provision.yaml.txt`) ; fichier placé dans un sous-dossier ou sur une autre partition que `SVA-BOOT` ; erreur de syntaxe YAML ; valeur invalide	Mettre la sonde hors tension, lire la carte sur un poste : présence et contenu de `provision-result.txt`, qui indique le motif du refus sans aucun secret	Corriger le fichier selon le compte rendu ; valider la syntaxe YAML (indentation par espaces, jamais de tabulation) ; recopier le fichier complet à la racine de la partition `SVA-BOOT` (ne modifier aucune autre partition de la carte), car un fichier refusé est supprimé ; redémarrer : le fichier est relu à chaque démarrage tant que la sonde n'est pas initialisée
Le fichier entier est refusé alors qu'une seule valeur est en cause	Comportement normal : la validation est atomique. Une seule erreur entraîne le refus de tout le fichier, et aucune section n'est appliquée	`provision-result.txt`	Corriger la valeur signalée et redéposer le fichier complet. Les sections absentes d'un fichier valide sont demandées par l'assistant de mise en service

Symptôme	Causes possibles	Vérifications	Actions correctives
Édition inattendue, ou sonde en mode restreint (aucun contrôle exécuté)	Carte d'une autre édition que celle commandée ; système copié sur une autre carte ; carte d'origine défectueuse ; carte de démonstration dont la période d'évaluation est terminée	Édition et cause affichées dans l'état de santé (`GET /api/v1/system`) ; numéro de série sur l'encart ou l'étiquette	Remettre la carte d'origine. Si elle est défectueuse, demander son remplacement (§ 20.6). Si l'édition ne correspond pas à la commande, contacter le service commercial (annexe L)
Mot de passe administrateur perdu	—	Existence d'un autre compte `admin` ; disponibilité du code de récupération (coffre-fort de l'organisation, chapitre 6, § 6.4.6) et du code d'installation (encart ou étiquette)	Un autre administrateur peut réinitialiser le mot de passe. Sinon, procédure de récupération du compte administrateur (chapitre 20, § 20.10.2) : dépôt de `recovery.txt` sur la partition `SVA-BOOT` , puis page de récupération ouverte pendant 30 minutes au démarrage, qui exige le code d'installation et le code de récupération. La configuration est conservée. Sans code de récupération, seule la réinitialisation usine reste possible
Page de récupération refusant les codes, ou fermée	Faute de saisie ; code de récupération déjà consommé par une récupération précédente (un nouveau code a alors été affiché) ; 3 essais épuisés pour cette ouverture ; délai de 30 minutes dépassé	Code de récupération le plus récent dans le coffre-fort ; code d'installation de l'encart ou de l'étiquette ; journal d'audit après récupération	Déposer de nouveau `recovery.txt` et redémarrer pour rouvrir la page ; saisir le code de récupération le plus récent. Sans code valide : réinitialisation usine (chapitre 20, § 20.10)
Réinitialisation usine physique (`factory-reset.txt`) sans effet	Comportement normal au premier redémarrage : la réinitialisation se confirme en deux temps et rien n'est effacé à la première étape ; ligne de confirmation absente ou mal saisie ; code de confirmation erroné ou expiré (validité de 24 heures)	Présence de `factory-reset-challenge.txt` sur la partition `SVA-BOOT` , qui contient le code aléatoire de 8 caractères ; code affiché sur la console HDMI ; journal d'audit	Écrire dans `factory-reset.txt` la ligne `EFFACER <code>` avec le code en cours de validité, puis redémarrer. Si le code a expiré, recommencer la première étape. Le numéro de série ne suffit jamais à autoriser la réinitialisation (chapitre 20, § 20.10)

Avertissement — Après un provisionnement réussi, vérifiez que `provision.yaml` a bien disparu de la partition `SVA-BOOT` , ainsi que les fichiers 802.1X désignés par `pkcs12File` et `caFile` s'il y en avait. Si l'un de ces fichiers subsiste, supprimez-le : il peut contenir des informations sensibles (empreinte du mot de passe, paramètres réseau et de proxy, mot de passe et clé privée de l'identifiant 802.1X).

19.5 Réseau

Symptôme	Causes possibles	Vérifications	Actions correctives
La sonde n'obtient pas d'adresse IP	Câble défectueux ; port désactivé ; contrôle d'accès réseau 802.1X sur le port ; serveur DHCP saturé	Voyants du port Ethernet ; journal du commutateur ; table DHCP	Changer de câble ou de port ; configurer l'authentification 802.1X de la sonde (chapitre 7), ou autoriser son adresse MAC dans le contrôle d'accès réseau (MAB) ; configurer une adresse statique
Échec de l'authentification 802.1X : la sonde reste sans accès au réseau	Certificat client expiré ou révoqué ; AC du serveur RADIUS différente de celle configurée (`dot1x.caRef`, ou `caFile` du provisionnement) ; identité (`dot1x.identity`) inconnue du serveur RADIUS ; port du commutateur non configuré pour 802.1X, ou dans un autre mode (PEAP, étiquetage VLAN, non pris en charge)	État de l'authentification affiché sur la console HDMI ; état de santé `network` dans `GET /api/v1/system` si la sonde reste joignable ; métrique `sva.dot1x.certificate.expiry` ; journaux du serveur RADIUS et du commutateur	La sonde réessaie automatiquement. Renouveler le certificat client (nouvelle demande générée sur la sonde ou import PKCS#12, chapitre 7) ; corriger l'AC ou l'identité ; faire configurer le port en EAP-TLS. Pour reprendre la main, raccorder temporairement la sonde à un port non contrôlé ou autorisé par adresse MAC (MAB)
Avertissement d'expiration du certificat 802.1X	Certificat client à moins de 30 jours de son expiration	`sva.dot1x.certificate.expiry` ; état de santé	Renouveler le certificat avant l'échéance (chapitre 7) : à expiration, la sonde perdrait l'accès au réseau
La sonde n'est pas joignable en IPv6 ; aucune adresse IPv6	IPv6 désactivé (valeur par défaut) ; mode `slaac` sans annonce de routeur sur le segment ; adresse ou passerelle statique erronée ; ICMPv6 (découverte des voisins) filtré	Section `network`, objet `ipv6` (`enabled`, `mode`, `address`, `gateway`) ; `ip -6 addr show eth0` et `ip -6 route` en SSH	Activer l'IPv6 et corriger ses paramètres (chapitre 7) ; autoriser ICMPv6 (découverte des voisins) sur le segment. L'IPv4 reste toujours actif et suffit à l'administration
Un contrôle vers une cible IPv6 utilise l'IPv4, ou l'inverse	Paramètre `ipFamily` à `auto` (valeur par défaut) : l'IPv6 n'est utilisé que si l'IPv6 de la sonde est activé **et** si la cible a une adresse AAAA ; sinon l'IPv4 est utilisé	`ipFamily` du contrôle ; contrôle `dns` avec `dns.recordType` à `AAAA` sur le nom de la cible	Fixer `ipFamily` à `ipv6` ou `ipv4` pour imposer la famille d'adresses (chapitre 8, § 8.1.3 ; chapitre 10) ; faire publier l'enregistrement AAAA

Symptôme	Causes possibles	Vérifications	Actions correctives
`dns_error` sur un contrôle dont `ipFamily` vaut `ipv6`	Le nom de la cible n'a pas d'adresse IPv6 (enregistrement AAAA absent)	Contrôle `dns` avec `dns.recordType` à `AAAA`	Faire publier l'enregistrement AAAA, ou revenir à `ipFamily: auto`
Contrôles en IPv6 en échec (`timeout`, `connection_refused`) alors qu'ils réussissent en IPv4	Enregistrement AAAA absent ou erroné ; pare-feu ou routage IPv6 incomplet entre la sonde et la cible ; règles de filtrage différentes entre IPv4 et IPv6	Contrôle `dns` avec `dns.recordType` à `AAAA` ; contrôle `tcp` ou `icmp` avec `ipFamily: ipv6` ; règles des pare-feu intermédiaires	Faire corriger l'enregistrement AAAA ; faire ouvrir les flux en IPv6 comme en IPv4 (annexe E). Le pare-feu local de la sonde applique les mêmes règles en IPv6 qu'en IPv4
SSH impossible à activer	SSH verrouillé définitivement (`ssh.locked: true`, par le fichier de provisionnement ou par `POST /api/v1/settings/ssh/lock`) ; compte sans rôle `admin`	Section `ssh` des paramètres (`locked`) ; journal d'audit	Le verrouillage est irréversible jusqu'à une réinitialisation usine. Transmettre une archive de diagnostic au support (§ 19.12) plutôt qu'un accès SSH. La récupération du compte administrateur reste possible
Session SSH fermée, SSH désactivé de lui-même	Comportement normal : l'échéance `enabledUntil` est atteinte (4 heures par défaut, 72 heures au maximum) ; les sessions ouvertes sont alors fermées	Section `ssh` (`enabled`, `enabledUntil`) ; journal d'audit (activation, connexions, désactivation)	Réactiver SSH avec une nouvelle échéance adaptée à l'intervention (chapitre 7, § 7.9.4)
Perte de l'accès après une modification de la configuration réseau	Erreur d'adresse, de masque ou de passerelle	Attendre 120 secondes : sans confirmation, la sonde revient automatiquement à la configuration précédente	Recommencer avec les bonnes valeurs ; confirmer dans le délai (chapitre 7)
Les noms ne sont pas résolus (`error.type = dns_error` sur de nombreux contrôles)	Serveurs DNS injoignables ou mal configurés ; nom interne non résolu par un DNS public	`getent hosts <nom>` en SSH ; contrôle `dns` de test (chapitre 10)	Corriger les serveurs DNS ; utiliser les serveurs DNS internes pour les noms internes
Heure incorrecte ; erreurs TLS généralisées (« certificat pas encore valide ») ; horodatages incohérents dans les plateformes	Synchronisation NTP impossible : flux UDP 123 bloqué, serveur NTP injoignable. Sans pile, l'horloge du Raspberry Pi 5 ne conserve pas l'heure hors tension	`timedatectl` ; état NTP dans `GET /api/v1/system`	Ouvrir le flux UDP 123 vers les serveurs NTP configurés ; utiliser le serveur NTP interne de l'entreprise
Accès Internet impossible malgré un proxy configuré	Adresse ou port du proxy erronés ; authentification refusée ; proxy n'autorisant pas `CONNECT` sur le port de destination	Message d'erreur des contrôles HTTP (`407 Proxy Authentication Required`) ; journaux du proxy	Corriger les paramètres (chapitre 7) ; autoriser l'adresse IP de la sonde ; ajouter les destinations internes dans la liste d'exclusions

Symptôme	Causes possibles	Vérifications	Actions correctives
Les ressources internes échouent alors que les ressources Internet fonctionnent	Requêtes internes envoyées au proxy	Liste d'exclusions du proxy	Ajouter les domaines et plages internes à la liste d'exclusions
L'interface d'administration est inaccessible depuis certains postes	Pare-feu entre le poste et la sonde ; liste d'adresses autorisées configurée sur la sonde	Test `curl -v https://<sonde>/health` depuis le poste ; paramètres de sécurité de la sonde	Ouvrir le port TCP 443 depuis le réseau d'administration ; ajuster la liste d'adresses autorisées
Les contrôles vers une zone réseau précise échouent tous (`connection_refused`, `timeout`)	Règles de pare-feu entre la zone de la sonde et la zone cible ; routage	`ping`, `curl -v`, contrôle `tcp` sur le port concerné	Faire ouvrir les flux nécessaires (annexe E)
Contrôles en erreur après déplacement de la sonde	Nouveau réseau, nouveau VLAN, nouveaux serveurs DNS ou proxy	Configuration réseau effective	Mettre à jour la configuration réseau, DNS et proxy

19.6 Contrôles techniques et applicatifs

19.6.1 Tableau des symptômes

Symptôme	Causes possibles	Vérifications	Actions correctives
`timeout`	Ressource lente ou surchargée ; flux filtré sans rejet explicite (le paquet est ignoré) ; timeout du contrôle trop court ; sonde saturée	Durées des exécutions précédentes ; `sva.agent.scheduler.lag` ; essai `curl -v` ou contrôle `tcp`	Corriger le flux réseau ; adapter `timeout` à un niveau réaliste (chapitre 8) ; réduire la charge de la sonde
`connection_refused`	Service arrêté ; mauvais port ; pare-feu qui rejette activement	Contrôle `tcp` sur l'hôte et le port	Vérifier la cible et le port ; vérifier l'état du service
`dns_error`	Nom inexistant, faute de frappe, DNS injoignable, enregistrement supprimé	Contrôle `dns` ; `getent hosts`	Corriger le nom ou la configuration DNS
`tls_error`	Certificat émis par une autorité non reconnue (autorité interne) ; nom du certificat ne correspondant pas à l'URL ; protocole TLS obsolète côté serveur ; proxy d'inspection TLS	`curl -v` (section TLS) ; détail de l'erreur dans l'exécution	Importer l'autorité de certification interne et la référencer dans le contrôle (`tls.caRef`, chapitre 10) ; utiliser le nom exact du certificat. Ne désactivez la vérification (`tls.verify: false`) qu'en dernier recours, et uniquement en environnement de test

Symptôme	Causes possibles	Vérifications	Actions correctives
`certificate_expired`	Certificat du serveur supervisé expiré	Date d'expiration dans le détail de l'exécution ; métrique `sva.tls.certificate.expiry`	Renouveler le certificat du serveur ; configurer `certificateExpiryWarningDays` pour être averti à l'avance
`http_status`	Code de réponse différent de l'attendu : redirection vers une page de connexion, 401, 403, 404, 5xx	Code reçu, en-tête `Location`	Corriger l'URL ou l'assertion ; ajouter l'authentification ; activer ou désactiver `followRedirects` selon le besoin
`assertion_failed` sur un contrôle HTTP ou API	Contenu modifié par une évolution de l'application ; assertion trop stricte ; réponse d'erreur au format attendu	Corps de la réponse, chemin JSONPath ou XPath	Mettre à jour l'assertion ; préférer des critères stables (chapitre 11)
`plugin_error` (état `UNKNOWN`)	Le plugin n'a pas pu s'exécuter jusqu'à son terme, ou a renvoyé un code retour en dehors de 0 à 3 : plugin introuvable ou non autorisé, non exécutable, arguments invalides, plugin interrompu	Sortie du plugin dans le détail de l'exécution	Vérifier le nom du plugin et ses arguments (chapitre 10) ; pour un plugin personnalisé, vérifier qu'il a été déposé et qu'il est exécutable
Plugin Nagios en état `UNKNOWN` en permanence	Dépendance manquante dans l'environnement isolé ; besoin de privilèges (certains plugins exigent des droits élevés)	Sortie du plugin	Utiliser un contrôle natif équivalent (`icmp`, `tcp`, `dns`, `http`) lorsqu'il existe ; les plugins exigeant des privilèges ne sont pas pris en charge
`database_error`	Hôte ou port injoignable ; nom de base erroné ; pilote incompatible avec la version du serveur ; requête invalide	Message d'erreur ; contrôle `tcp` sur le port du SGBD (5432, 3306, 1433, 1521…)	Corriger les paramètres ; vérifier les versions de SGBD prises en charge, indiquées dans les notes de version
`auth_failed`	Mot de passe changé ou expiré ; compte verrouillé après trop d'échecs ; compte désactivé	Date de dernière rotation du secret ; état du compte côté application	Mettre à jour le secret (chapitre 9) ; demander un compte de test dont le mot de passe n'expire pas ou dont la rotation est planifiée
Règle de base de données en échec alors que les données semblent correctes	Portée de la règle mal choisie (`allRows`, `anyRow`, `firstRow`) ; type de valeur (texte comparé à un nombre) ; nom de colonne en majuscules ou minuscules selon le SGBD	Résultat brut de la requête dans le détail de l'exécution	Ajuster la portée, le type ou l'alias de colonne (chapitre 11)

Symptôme	Causes possibles	Vérifications	Actions correctives
Résultats incohérents d'une exécution à l'autre	Répartition de charge vers des serveurs hétérogènes ; cache ; données de test modifiées par d'autres utilisateurs ; dépendance à l'heure	Historique, attributs des requêtes, heure des échecs	Stabiliser les données de test ; superviser chaque serveur individuellement si nécessaire ; utiliser des assertions moins dépendantes du contexte
Bagotage (alternance rapide OK / CRITICAL)	Ressource réellement instable ; seuil de performance trop proche des valeurs habituelles ; réseau intermittent	Distribution des durées ; corrélation avec l'heure ou la charge	Configurer des nouvelles tentatives (`retry.attempts` et `retry.interval`) pour confirmer un échec ; ajuster `warningDuration` ; signaler l'instabilité aux équipes concernées
État `STALE` sur un contrôle	Le contrôle n'a produit aucun résultat depuis deux intervalles : sonde saturée, plan d'exécution bloqué par un contrôle plus long que prévu. Un contrôle n'est jamais `STALE` en dehors de la fenêtre d'activité de son plan, ni pendant l'état de maintenance de la sonde et l'intervalle qui suit la reprise	`sva.agent.scheduler.lag` ; durée du plan (`maxDuration`) ; `activeWindows`	Réduire la charge ou la fréquence ; ajuster les timeouts du plan ; vérifier les fenêtres d'activité (chapitre 12)
Contrôle jamais exécuté	Contrôle désactivé ; `schedule: null` sans rattachement à un plan ; plan désactivé ; édition ne comprenant pas ce type de contrôle	État du contrôle et des plans ; édition dans `GET /api/v1/system`	Activer le contrôle ; le rattacher à un plan ; vérifier l'édition
Erreur `resource_exhausted`	Mémoire ou nombre de processus limité atteint dans l'environnement isolé	`free -h` ; métriques de la sonde	Réduire la concurrence ; espacer les exécutions lourdes ; vérifier la RAM de la carte par rapport à l'édition

19.6.2 Distinguer un problème de la sonde d'un problème de la ressource

Lorsqu'un contrôle échoue, posez-vous les questions suivantes, dans l'ordre :

1. **La ressource est-elle accessible depuis un poste situé dans le même réseau que la sonde ?** Si non, le problème n'est pas propre à la sonde.
2. **D'autres contrôles de la même sonde échouent-ils au même moment ?** Si tous échouent, examinez la sonde et son réseau (§ 19.3 et § 19.5).
3. **Une autre sonde, sur un autre site, supervise-t-elle la même ressource ?** Si elle la voit en `OK`, le problème est lié au chemin réseau du premier site.
4. **L'exécution d'essai reproduit-elle l'échec ?** Si non, l'échec était probablement transitoire. Les nouvelles tentatives confirmeront ou non le diagnostic.

19.7 Scénarios navigateur (Puppeteer)

Les scénarios navigateur sont les contrôles les plus riches, et donc les plus exposés aux évolutions des applications. Le chapitre 16 décrit les bonnes pratiques qui préviennent la plupart des incidents ci-dessous.

19.7.1 Tableau des symptômes

Symptôme	Causes possibles	Vérifications	Actions correctives
Tous les scénarios en `UNKNOWN` avec `browser_crash`	Chromium ne démarre pas : mémoire insuffisante, mise à jour incomplète, environnement isolé défaillant	`sva.browser.launches` ; `free -h` ; `journalctl -u sva-runner` ; dernière mise à jour appliquée	Redémarrer la sonde ; vérifier la RAM (8 Go pour l'édition Enterprise) ; si le problème suit une mise à jour, contacter le support
`browser_crash` occasionnel sur un scénario précis	Page très lourde (vidéos, cartes, nombreux scripts tiers) ; fuite de mémoire de la page	Taille et contenu de la page ; durée de l'exécution	Bloquer les domaines tiers inutiles (`blockDomains`) ; réduire la taille de la fenêtre (`viewport`) ; ne pas exécuter deux scénarios lourds en parallèle
`resource_exhausted` sur les scénarios	Concurrence navigateur trop élevée pour la RAM ; scénarios trop longs	Concurrence configurée ; métriques mémoire	Revenir à une seule exécution navigateur simultanée ; regrouper les scénarios dans un plan d'exécution séquentiel
`element_not_found`	Sélecteur devenu invalide après une évolution de l'interface ; élément affiché plus tard que prévu ; élément situé dans une iframe ; page différente (bannière de consentement, page de maintenance)	Capture d'écran de l'échec ; comparaison avec la page actuelle dans un navigateur	Mettre à jour le sélecteur en privilégiant un attribut stable (`data-testid` , rôle, texte) ; traiter la bannière de consentement en début de scénario ; préciser la propriété `frame`
`timeout` sur une attente d'élément	Application lente ; attente d'un événement qui ne se produit jamais (navigation attendue alors que la page se met à jour sans changer d'URL)	Durée des étapes précédentes ; capture	Remplacer l'attente de navigation par l'attente d'un élément ; ajuster le timeout de l'étape
`navigation_error`	URL injoignable depuis la sonde ; erreur DNS ou TLS dans le navigateur ; redirection vers un domaine non autorisé par le pare-feu ou le proxy de l'entreprise	Message d'erreur (code `net::ERR_…` renvoyé par Chromium)	Traiter comme un incident réseau (§ 19.5, § 19.6) ; faire autoriser la destination dans le pare-feu ou le proxy de l'entreprise (chapitre 7)
`script_error`	Erreur de programmation dans un scénario scripté (variable non définie, syntaxe) ; module non autorisé	Message et ligne de l'erreur	Corriger le script ; tester sur poste de travail avec le lanceur local (chapitre 14) puis par une exécution d'essai
`assertion_failed` sur un texte	Libellé modifié ; espaces ou casse différents ; texte traduit selon la langue du navigateur	Capture ; texte réel de l'élément	Assouplir l'assertion (contient plutôt qu'égal) ; fixer la langue attendue

Symptôme	Causes possibles	Vérifications	Actions correctives
Le scénario fonctionne sur le poste de développement mais pas sur la sonde	Résolution DNS ou proxy différents ; autorité de certification interne absente de la sonde ; écran de taille différente ; langue ou fuseau horaire différents ; poste déjà authentifié (cookies), alors que la sonde démarre avec un profil vierge ; version de Chromium différente	Comparer le contexte : réseau, `viewport`, langue, cookies	Tester sur le poste avec un profil vierge (lanceur local) ; aligner `viewport` ; ajouter l'étape d'authentification ; importer l'autorité de certification interne
Le scénario échoue sur une page de vérification anti-robot, un CAPTCHA ou une double authentification	Protection anti-automatisation de l'application ; politique de sécurité imposant une authentification multifacteur	Capture d'écran	Ne pas chercher à contourner ces protections. Demander aux équipes de l'application une exemption pour l'adresse IP de la sonde ou un compte de test dédié sans authentification multifacteur, dans un environnement prévu à cet effet
Les mesures de durée varient fortement	Scripts tiers (publicité, statistiques) ; cache vide à chaque exécution (comportement attendu : profil vierge) ; charge de la sonde	Durée par étape ; requêtes réseau	Bloquer les domaines tiers non pertinents ; ne comparer que des exécutions de même nature
Captures d'écran absentes	Option `captureOnFailure` désactivée ; quota d'artefacts atteint ; échec survenu avant l'ouverture de la page	Paramètres du contrôle ; `sva.storage.usage` (zone `artifacts`)	Réactiver l'option ; ajuster la rétention des artefacts (chapitre 9)
Scénario en `CRITICAL` à cause d'une étape intermédiaire alors que l'objectif est atteint	Étape de vérification trop stricte ou devenue inutile	Détail des étapes	Réviser le scénario avec l'équipe applicative

19.7.2 Lire un échec de scénario

Pour un scénario en échec, le détail de l'exécution fournit quatre éléments complémentaires :

1. **L'étape en échec**, nommée par le scénario : elle situe le problème dans le parcours.
2. **Le type d'erreur** : il indique la nature du problème (élément absent, délai dépassé, assertion non vérifiée).
3. **Le message** : il précise le sélecteur, la valeur attendue ou l'erreur de navigation.
4. **La capture d'écran et le journal de console** : ils montrent ce que voyait réellement le navigateur et les erreurs JavaScript de la page.

> ✓ **Conseil** — Une capture d'écran qui montre une page d'erreur de l'application (« Service indisponible », erreur 500) signifie que le scénario a correctement détecté un incident applicatif. Une capture qui montre une page normale mais un élément introuvable oriente plutôt vers un scénario à mettre à jour.

19.8 Destinations d'export et intégrations

19.8.1 Tester une destination

Avant toute analyse, lancez le test intégré de la destination concernée. Il envoie un point de test et renvoie le code HTTP, le corps de la réponse d'erreur et la latence.

```bash
curl --cacert sonde-ca.pem -X POST \
     -H @"$HOME/.sva/entete-auth" \
     https://sonde-paris-01.example.internal/api/v1/exporters/dynatrace/test
```

19.8.2 Tableau des symptômes

Ce tableau reprend, dans la démarche de diagnostic de ce chapitre, les cas du tableau de dépannage des exports du chapitre 18 (§ 18.16).

Symptôme	Causes possibles	Vérifications	Actions correctives
HTTP 401	Clé ou jeton d'ingestion absent, invalide ou révoqué ; mauvais format d'en-tête (`Api-Token` pour Dynatrace, en-tête de clé d'API pour Datadog)	Test de la destination ; secret référencé dans les en-têtes	Mettre à jour le secret ; vérifier le format exact de l'en-tête (chapitre 18)
HTTP 403	Jeton valide mais sans les droits d'ingestion requis ; pour Datadog, endpoint ne correspondant pas au « site » du compte	Portées du jeton Dynatrace (ingestion de métriques, logs, traces) ; site Datadog	Ajouter les portées nécessaires ; utiliser l'endpoint du bon site Datadog
HTTP 400 sur les métriques vers Dynatrace ou l'intake direct Datadog	**Temporalité cumulative** envoyée à une plateforme qui n'accepte que le **delta**	Paramètre `metrics.temporality` de la destination	Régler `metrics.temporality` sur `delta` (préréglages `dynatrace` et `datadog-agentless`)
Succès partiel signalé, certaines métriques absentes dans Dynatrace	Requête mêlant métriques acceptées et refusées (type non pris en charge, temporalité)	Journal de la sonde (`sva.export.failures`)	Utiliser le préréglage `dynatrace`, qui fixe la temporalité delta et des histogrammes à limites explicites
HTTP 404	Chemin d'endpoint incorrect, notamment pour le récepteur OTLP de Prometheus (`/api/v1/otlp/v1/metrics`)	URL complète testée	Pour Prometheus, renseigner l'URL complète dans `signalEndpoints.metrics`
HTTP 413	Lot trop volumineux	Taille des lots (1 000 points par lot par défaut)	Réduire le volume (désactiver `metrics.stepHistograms`, allonger `metrics.exportInterval`) ; si l'erreur persiste, contacter le support
HTTP 429	Limitation de débit de la plateforme ; quota d'ingestion atteint	En-tête `Retry-After` ; consommation dans la plateforme	La sonde réessaie automatiquement ; réduire le volume (désactiver les histogrammes par étape, espacer l'export) ; ajuster le quota

Symptôme	Causes possibles	Vérifications	Actions correctives
HTTP 5xx, erreurs réseau	Plateforme indisponible ; proxy ; pare-feu	Test de la destination ; connectivité vers l'endpoint	Vérifier le flux sortant (annexe E) ; la sonde réessaie pendant `retry.maxElapsed` puis abandonne le lot (`sva.export.dropped`)
Prometheus ne collecte pas la sonde (cible `DOWN`)	Pare-feu ; jeton absent ou de mauvais rôle ; certificat non reconnu par Prometheus	Page des cibles de Prometheus (message d'erreur) ; `curl` vers `/metrics` avec le jeton	Ouvrir le port 443 depuis Prometheus ; utiliser un jeton de rôle `metrics` ; renseigner `tls_config.ca_file`
Données reçues par le récepteur OTLP de Prometheus mais invisibles	Récepteur non activé (`--web.enable-otlp-receiver`) ; ingestion hors ordre refusée ; noms traduits avec suffixes	Métriques internes de Prometheus sur la traduction OTLP	Activer le récepteur ; configurer `out_of_order_time_window` ; rechercher les noms traduits (`sva_check_duration_seconds…`)
Métriques présentes mais aucun tableau de bord ni alerte	Comportement normal : l'ingestion OTLP ne crée ni tableau de bord ni alerte	—	Importer les modèles fournis par l'éditeur (`exemples/grafana/sva-tableau-de-bord.json` pour Grafana, `exemples/prometheus/sva-regles-alerte.yml` pour les règles d'alerte Prometheus) ou créer les tableaux de bord et alertes (chapitre 18)
Nagios ou Centreon affiche `UNKNOWN` pour `check_sva`	Sonde injoignable depuis le serveur Nagios ; jeton invalide ; identifiant de contrôle erroné ; texte `MAINTENANCE` (code 3) : la sonde est en état de maintenance pour l'activation d'une mise à jour, situation normale et temporaire (§ 19.10)	Exécuter le plugin en ligne de commande sur le serveur Nagios	Corriger le jeton (rôle `metrics`), l'identifiant du contrôle (option `--check`) et le flux HTTPS
Trous dans les courbes après une coupure réseau	Comportement documenté : les métriques ne sont pas rejouées après une indisponibilité de la destination	`sva.export.dropped`	L'historique des résultats reste disponible sur la sonde, dans la limite de sa rétention locale (chapitre 17). Pour réduire les pertes, utiliser un collecteur OpenTelemetry avec file persistante (chapitre 18) : voir l'encadré ci-dessous sur sa portée
Absence de données de la sonde dans la plateforme alors qu'elle fonctionne	Export interrompu (voir lignes précédentes) ; heure de la sonde décalée ; activation d'une mise à jour en cours (absence normale pendant le redémarrage)	Test de la destination ; `timedatectl`	Corriger l'export ; corriger la synchronisation NTP

> **Information** — Un collecteur OpenTelemetry à file persistante **réduit les pertes** de données ; il ne garantit pas leur livraison. Distinguez les deux segments du trajet :
>
> - **de la sonde au collecteur** : la sonde réessaie l'envoi pendant `retry.maxElapsed`, puis abandonne le lot. Un lot abandonné n'atteint jamais le collecteur, qui ne peut préserver que les données qu'il a reçues. Ces abandons sont observables par la métrique `sva.export.dropped` ;
> - **du collecteur à la destination** : la file persistante conserve les données reçues pendant une indisponibilité de la destination, dans la limite de sa taille et de sa durée de conservation. Au-delà, le collecteur abandonne à son tour des données ; surveillez ses propres métriques de file et d'abandon.
>
> Placer le collecteur au plus près de la sonde, sur un réseau fiable, réduit le risque sur le premier segment. En cas de perte, l'historique local de la sonde reste la référence, dans la limite de sa rétention (chapitre 17).

19.9 Stockage

Symptôme	Causes possibles	Vérifications	Actions correctives
Alerte d'espace disque, `sva.storage.usage` élevé	Rétention trop longue pour le nombre de contrôles ; nombreux artefacts ; carte de faible capacité	`df -h` ; paramètres de rétention	Réduire la rétention des exécutions détaillées ou des artefacts (chapitre 9) ; la sonde purge automatiquement les données les plus anciennes avant saturation
Disque plein	Purge insuffisante ; fichiers anormaux	`df -h`	La sonde poursuit les contrôles et supprime en priorité les artefacts les plus anciens ; si la situation persiste, contacter le support
Message signalant une récupération de la base au démarrage, historique des résultats vide	La sonde a détecté une base corrompue (généralement après une coupure électrique) et l'a restaurée à partir de la dernière sauvegarde de configuration	Journal d'audit et métrique `sva.storage.recoveries`	La configuration est préservée, l'historique récent est perdu. Identifier la cause de la coupure (onduleur, alimentation) ; envisager le remplacement de la carte si l'incident se répète
La sonde démarre en mode « non initialisée » après un incident	Aucune sauvegarde de configuration valide n'a pu être restaurée	Interface : message d'incident	Importer le dernier export de configuration (chapitre 20, § 20.5) ou relancer la mise en service
Erreurs en lecture seule (« read-only file system »)	Le système a remonté une partition en lecture seule après des erreurs : carte défaillante	`dmesg` ; `mount`	Exporter la configuration si l'interface répond encore ; remplacer la carte (chapitre 20, § 20.6)

Symptôme	Causes possibles	Vérifications	Actions correctives
État de santé `storage` en `CRITICAL` sur une sonde équipée d'un SSD NVMe	SSD absent, mal enfiché, non détecté ou défaillant ; carte M.2 HAT+ mal raccordée ; alimentation insuffisante (l'alimentation officielle de 27 W est requise)	État de santé `storage` dans `GET /api/v1/system` ; `dmesg --level=err,warn` (erreurs `nvme` ou PCIe) ; `vcgencmd get_throttled`	Les contrôles continuent : l'historique local est conservé sur la microSD en mode réduit, et la configuration, les secrets et le système, qui résident sur la microSD, ne sont pas affectés. Arrêter proprement la sonde, vérifier le raccordement de la carte M.2 HAT+ et du SSD, puis redémarrer. Si le SSD est défaillant, le remplacer par un modèle de la liste des modèles pris en charge, publiée dans les notes de version, et l'activer de nouveau (chapitre 20)
Historique plus court que prévu, artefacts moins nombreux	Fonctionnement en mode réduit sur la microSD après la perte du SSD	État de santé `storage`	Rétablir ou remplacer le SSD (ligne précédente). L'historique conservé sur le SSD défaillant peut être perdu

19.10 Mises à jour

Symptôme	Causes possibles	Vérifications	Actions correctives
Aucune mise à jour proposée	Mode manuel sans vérification récente ; dépôt injoignable (proxy, pare-feu) ; période de support terminée (`supportEndsAt` dépassée : plus aucun correctif, état de santé `support` en `WARNING`) ; pour une version fonctionnelle seulement, justificatif de maintenance absent ou échu (`maintenanceEndsAt`) : les correctifs de sécurité restent proposés sans justificatif	État des mises à jour (`GET /api/v1/updates`, champs `supportEndsAt` et `maintenanceEndsAt`) ; flux HTTPS vers le dépôt (`https://updates-synovex.incitypark.fr` par défaut, ou votre miroir interne)	Ouvrir le flux sortant ; vérifier le proxy ; importer un justificatif de maintenance en cours de validité (`PUT /api/v1/system/maintenance`, chapitre 20) pour accéder aux versions fonctionnelles
Image refusée alors que sa signature est valide	Métadonnées du dépôt ou du miroir expirées (`validUntil`, 7 jours) : miroir interne non synchronisé ; image de version inférieure à la version installée (règle anti-retour en arrière, y compris pour un import hors ligne)	Message d'erreur ; date de dernière synchronisation du miroir ; version installée dans `GET /api/v1/updates`	Synchroniser le miroir ou revenir au dépôt de l'éditeur ; importer une image de version égale ou supérieure (chapitre 20, § 20.4.4 et § 20.4.5)
Activation programmée refusée (`422`)	Date `scheduledAt` passée ou située à moins d'une minute	Corps de la requête ; heure du poste (dates en UTC)	Indiquer une date future, à plus d'une minute ; une date antérieure à l'expiration du préavis est acceptée avec `noticeReduced: true` (chapitre 20, § 20.4.2)
Échec de vérification de signature	Image de mise à jour (*bundle*) altérée ou incomplète, téléchargée ou téléversée (`POST /api/v1/updates/upload`) ; heure système incorrecte	`timedatectl` ; message d'erreur	Ne pas forcer l'installation ; corriger l'heure ; relancer le téléchargement ; contacter le support si l'erreur persiste
Les scénarios échouent après une mise à jour	Nouvelle version de Chromium ou de Puppeteer modifiant un comportement	Notes de version ; comparaison avant/après	Consulter les notes de version ; adapter les scénarios concernés ; signaler la régression au support

Symptôme	Causes possibles	Vérifications	Actions correctives
La sonde ne redémarre pas correctement après une mise à jour	Coupure pendant l'installation ; incident logiciel de la nouvelle version ; blocage du système avant le démarrage des services	Accès à `/health` ; écran HDMI	Patienter sans couper l'alimentation : l'autotest dispose de 10 minutes ; s'il échoue, la sonde revient automatiquement à l'emplacement système précédent (ligne suivante). Ces 10 minutes ne sont pas une durée garantie de retour : si le système est bloqué, le retour suppose un redémarrage effectif. Un blocage avant le démarrage des services est traité par le chien de garde du chargeur d'amorçage puis par le chien de garde matériel, qui réinitialisent la carte (chapitre 20, § 20.4.5) ; toute réinitialisation pendant le démarrage d'essai ramène la sonde sur l'emplacement précédent, car l'indicateur *tryboot* est à usage unique (chapitre 20, § 20.4.5). Si la sonde reste injoignable au-delà, contacter le support
Mise à jour annulée par retour arrière automatique : la version précédente est de nouveau active	La nouvelle version n'a pas été jugée saine dans les 10 minutes suivant son démarrage (moteur principal, API ou autotest en échec) ; la sonde est revenue à l'emplacement précédent et a restauré les bases sauvegardées avant la migration	`GET /api/v1/updates` : version de chaque emplacement et emplacement actif ; journal d'audit (installation, redémarrage, retour arrière)	Aucune action urgente : la sonde fonctionne sur la version précédente, avec sa configuration. Ne pas relancer la même mise à jour en boucle ; préparer une archive de diagnostic et contacter le support (§ 19.12)
Mise à jour activée en dehors de la période souhaitée	Fenêtres de maintenance mal réglées : jours, heure de début ou durée erronés, ou fuseau horaire de la sonde différent de celui retenu pour définir les fenêtres ; correctif critique activé dès que possible (`updates.security.criticalPolicy: asap`) ; activation déclenchée ou programmée par un administrateur (`POST /api/v1/updates/apply`, éventuellement avec `scheduledAt`)	Section `updates` des paramètres : `windows` (`days`, `start` au format `HH:MM`, `duration`) et `security.criticalPolicy` ; fuseau horaire de la sonde (chapitre 7) ; journal d'audit ; événement `update.scheduled`	Corriger les fenêtres ou le fuseau horaire ; revenir à `criticalPolicy: next-window` si nécessaire ; passer un canal en mode `manual` si votre organisation l'exige (chapitre 20)

Symptôme	Causes possibles	Vérifications	Actions correctives
Mise à jour installée mais qui ne s'active pas (état de santé `updates` en `WARNING` ou `CRITICAL`)	Aucune fenêtre de maintenance définie : sans fenêtre, aucune activation automatique ordinaire ni aucun correctif critique en mode `next-window` n'est activé ; activation reportée (`POST /api/v1/updates/postpone`, jusqu'à la date `until`) ; préavis non respecté : une activation automatique ordinaire doit être annoncée au moins `updates.notice` avant son début (24 heures par défaut), un correctif critique au moins 15 minutes avant le début de la fenêtre retenue, sinon l'activation passe à la fenêtre suivante ; canal concerné en mode `manual`	`GET /api/v1/updates` (champ `scheduledActivation`) ; métriques `sva.update.pending` , `sva.maintenance.next_start` et `sva.maintenance.next_end` ; section `updates` (`windows` , `notice` , `security.mode` , `security.criticalPolicy` , `feature.mode`) ; journal d'audit (report) ; événement `update.cancelled` (motif de l'annulation)	Définir au moins une fenêtre de maintenance ; ou activer la mise à jour par `POST /api/v1/updates/apply` , immédiatement ou à une heure donnée (`scheduledAt`) ; vérifier l'échéance d'un report ; pour un correctif critique (`CRITICAL`), ne pas différer l'activation au-delà du nécessaire : le report est limité à 7 jours (chapitre 20)
Contrôles en `UNKNOWN` avec le type d'erreur `maintenance`	Exécutions interrompues par le drainage au début de l'activation d'une mise à jour : elles ne se sont pas terminées dans le délai `updates.drainTimeout` (5 minutes par défaut)	Heure des exécutions comparée aux fenêtres de maintenance ; événements `maintenance.started` et `maintenance.ended` ; journal d'audit	Aucune action si le phénomène est ponctuel et coïncide avec une activation. S'il touche régulièrement les mêmes plans ou scénarios longs, allonger `drainTimeout` (30 minutes au maximum) ou déplacer les fenêtres hors des créneaux de ces plans (chapitre 12)
Alertes reçues pendant la maintenance de la sonde (sonde injoignable, absence de données de la sonde dans la plateforme)	L'état de maintenance ne suffit pas à supprimer les alertes : la sonde est injoignable pendant son redémarrage ; périodes de maintenance non déclarées dans la plateforme, ou non alignées sur les fenêtres de la sonde	Heure des alertes comparée à `sva.maintenance.next_start` et `sva.maintenance.next_end` ; métrique `sva.agent.maintenance`	Configurer dans la plateforme des périodes de maintenance alignées sur les fenêtres de la sonde (Alertmanager, Datadog, Dynatrace, Centreon, Nagios) et, pour Prometheus, exclure la sonde en maintenance des règles d'alerte (chapitre 18) ; pour une supervision sans interruption, déployer plusieurs sondes par site avec des fenêtres qui ne se chevauchent pas (chapitre 20)

19.11 Types d'erreur et actions de premier niveau

Le tableau suivant reprend la liste fermée des types d'erreur que la sonde associe à une exécution (champ `error.type`). L'annexe H en donne une description complète.

Type d'erreur	État habituel	Origine probable	Première action
`timeout`	`CRITICAL`	Ressource lente ou flux filtré	Vérifier la ressource et le flux ; ajuster le timeout
`dns_error`	`CRITICAL`	Nom ou DNS	Contrôle `dns` ; configuration DNS
`connection_refused`	`CRITICAL`	Service arrêté, mauvais port	Contrôle `tcp`
`connection_reset`	`CRITICAL`	Coupure par le serveur, un pare-feu ou un équipement intermédiaire	Journaux du serveur et des équipements réseau

Type d'erreur	État habituel	Origine probable	Première action
`tls_error`	`CRITICAL`	Autorité, nom ou protocole TLS	Autorité de certification interne, nom exact
`certificate_expired`	`CRITICAL`	Certificat serveur expiré	Renouvellement du certificat
`http_status`	`CRITICAL`	Code HTTP inattendu	Code, redirections, authentification
`assertion_failed`	`CRITICAL`	Contenu différent de l'attendu	Comparer réel et attendu
`element_not_found`	`CRITICAL`	Sélecteur obsolète, élément tardif	Capture d'écran ; sélecteur
`navigation_error`	`CRITICAL`	Navigation impossible	Réseau ; URL
`script_error`	`CRITICAL`	Erreur dans un scénario scripté	Corriger et tester le script
`browser_crash`	`UNKNOWN`	Chromium défaillant	Mémoire, redémarrage
`plugin_error`	`UNKNOWN`	Plugin Nagios qui n'a pas pu s'exécuter jusqu'à son terme, ou code retour en dehors de 0 à 3	Sortie du plugin, nom et arguments
`secret_missing`	`UNKNOWN`	Secret référencé par le contrôle mais absent de la sonde (non créé, supprimé, mal orthographié, configuration importée sans ses secrets)	Un administrateur crée le secret, avec ses hôtes autorisés (chapitre 9, § 9.8)
`database_error`	`CRITICAL`	Connexion ou requête SQL	Paramètres, pilote, requête
`no_data`	`UNKNOWN` (selon `onEmpty`)	Requête `database` sans aucune ligne renvoyée	Réécrire la requête pour qu'elle renvoie toujours une ligne (chapitre 11)
`auth_failed`	`CRITICAL`	Identifiants refusés	Secret, état du compte
`resource_exhausted`	`UNKNOWN`	Sonde à court de ressources	Concurrence, RAM
`runner_crash`	`UNKNOWN`	Moteur de scénarios interrompu	Redémarrage automatique ; si répété, support
`agent_restart`	`UNKNOWN`	Moteur principal redémarré pendant l'exécution	Cause du redémarrage (mise à jour, coupure)
`maintenance`	`UNKNOWN`	Exécution interrompue par le drainage au début de l'activation d'une mise à jour	Aucune si ponctuel ; sinon ajuster `updates.drainTimeout` ou les fenêtres (§ 19.10)
`unknown`	`UNKNOWN`	Erreur non classée	Message ; support si répété

19.12 Préparer une demande au support

Une demande complète réduit fortement le délai de résolution. Avant de contacter le support (coordonnées en annexe L), rassemblez les informations suivantes.

Information	Où la trouver
Version du produit, édition, version de Chromium et de Puppeteer	`GET /api/v1/system` ou page de santé de l'interface
Numéro de série ou adresse MAC de la sonde	Étiquette du boîtier complet ou `GET /api/v1/system`

Information	Où la trouver
Offre matérielle (microSD préconfigurée ou boîtier complet), modèle et RAM du Raspberry Pi 5, alimentation utilisée	Inventaire
Description du symptôme, date et heure (UTC) de début, fréquence	Votre constat, historique des exécutions
Identifiants des exécutions concernées (`runId`) et des contrôles (`checkId`)	Interface, `GET /api/v1/runs`
Détail des exécutions en échec (étapes, type d'erreur, message)	`GET /api/v1/runs/{runId}`
Captures d'écran et journaux de console des scénarios en échec	Artefacts (après vérification de leur contenu, voir ci-dessous)
Résultat du test de la destination d'export, le cas échéant	`POST /api/v1/exporters/{id}/test`
Résultat de `vcgencmd get_throttled` et température	SSH ou métriques
Modifications récentes : configuration, réseau, mise à jour, évolution de l'application supervisée	Journal d'audit, gestion des changements
Actions déjà tentées et leur effet	Votre historique
Archive de diagnostic : version, état de santé, configuration sans les secrets et journaux récents	`POST /api/v1/system/diagnostics` (rôle `admin`) : téléchargez l'archive, vérifiez son contenu (captures, journaux), puis transmettez-la vous-même au support par le canal qu'il vous indique

Avertissement — Ne transmettez jamais de secret au support : mots de passe des comptes de test, jetons d'API, clés d'ingestion, empreintes de mot de passe, clés privées. Avant de joindre une capture d'écran, un journal de console ou un export de configuration, vérifiez qu'il ne contient ni donnée personnelle ni donnée sensible. Les exports de configuration doivent être produits **sans les secrets** (chapitre 20, § 20.5).

Attention — Avant de transmettre l'archive de diagnostic, ouvrez-la et vérifiez son contenu. Les journaux et les captures d'écran peuvent contenir des données personnelles ou un secret réaffiché par une application supervisée : le masquage automatique des valeurs connues ne couvre pas ces cas. Retirez ou masquez tout élément sensible. Transmettez l'archive uniquement par le canal indiqué par le support. La durée de conservation des archives par l'éditeur est définie dans sa politique de support, selon les conditions de votre contrat, et indiquée sur le portail du support (`https://support-synovex.incitypark.fr`).

Information — Modalités d'intervention à distance du support :

- L'éditeur ne dispose d'**aucun accès permanent** à la sonde : aucun compte, aucune clé ni aucun canal ne lui est réservé.
- La sonde n'envoie **aucune donnée** à l'éditeur (pas de télémétrie). Ses seuls échanges avec l'éditeur sont la vérification et le téléchargement des mises à jour, si vous les avez activés.
- Le mode d'intervention privilégié est l'**archive de diagnostic**, que vous produisez (`POST /api/v1/system/diagnostics`), vérifiez et transmettez vous-même au support.
- Si une intervention en ligne de commande est nécessaire, seul un administrateur de votre organisation peut activer un **accès SSH temporaire** (§ 19.2.4 ; chapitre 7, § 7.9.4) : échéance obligatoire, de 4 heures par défaut et de 72 heures au maximum, désactivation automatique à l'échéance avec fermeture des sessions, et enregistrement de l'activation, des connexions et de la désactivation dans le journal d'audit. Déposez la clé publique fournie par le support et désactivez SSH dès la fin de l'intervention.
- Si SSH a été verrouillé (`ssh.locked`), aucune intervention SSH n'est possible : le support travaille alors à partir de l'archive de diagnostic.

Les conditions contractuelles d'intervention (délais, horaires) sont définies dans votre contrat de maintenance.

Chapitre 20 — Maintenance et mises à jour

Une sonde de supervision fonctionne en continu, sans surveillance directe, souvent pendant plusieurs années. Sa fiabilité dépend de quelques opérations simples mais régulières : arrêter l'appareil proprement, protéger son support de stockage, appliquer les mises à jour, sauvegarder la configuration, renouveler les secrets et adapter les scénarios aux évolutions des applications.

Ce chapitre décrit ces opérations et propose, au § 20.11, un plan de maintenance préventive.

> **Information** — Les exemples de ce chapitre utilisent des noms et adresses fictifs (`sonde-paris-01.example.internal`) et suivent la convention du chapitre 9 (§ 9.3.3) : la variable `SONDE` contient l'adresse de la sonde, la variable `CA` le chemin de son certificat, récupéré et vérifié selon le chapitre 6 (§ 6.4.4), et la fonction `sapi` appelle l'API avec ce certificat et le jeton lu dans le fichier d'en-tête protégé `$HOME/.sva/entete-auth`. Le jeton n'apparaît ainsi jamais dans la ligne de commande.

20.1 Répartition des responsabilités

Domaine	Éditeur	Client
Système d'exploitation, moteur principal, moteur de scénarios, Chromium	Publie des images système signées et validées ; publie les correctifs de sécurité pendant toute la période de support (§ 20.4.6)	Choisit le mode de chaque canal, définit les fenêtres de maintenance et le préavis, active ou laisse activer les mises à jour (§ 20.4.2)
Matériel	Garantie du boîtier complet, selon les conditions de votre contrat	Environnement d'installation, alimentation, refroidissement ; matériel fourni par le client dans l'offre microSD préconfigurée
Configuration (contrôles, plans, destinations)	—	Conception, sauvegarde, évolution
Scénarios	Exemples et documentation	Maintenance au rythme des évolutions applicatives
Secrets et comptes de test	—	Création, rotation, révocation
Sauvegardes	Sauvegarde automatique locale de la configuration	Export régulier hors de la sonde

20.2 Arrêt et redémarrage

20.2.1 Pourquoi un arrêt propre est indispensable

La sonde écrit en permanence sur son support de stockage : résultats, historique, journaux d'exécution. Couper l'alimentation sans arrêt préalable interrompt ces écritures. Les mécanismes de la sonde (base de données transactionnelle, écritures atomiques des fichiers de configuration, récupération automatique au démarrage) limitent fortement le risque, mais ne l'éliminent pas. Des coupures répétées peuvent abîmer le système de fichiers ou la carte microSD elle-même.

> **Avertissement** — Ne débranchez jamais l'alimentation d'une sonde en fonctionnement, sauf en cas d'urgence (fumée, odeur de brûlé, liquide renversé). Arrêtez d'abord la sonde, puis attendez l'extinction complète avant de débrancher.

20.2.2 Arrêter la sonde

Objectif : mettre la sonde hors tension sans risque pour les données.

Prérequis : compte de rôle `admin` (arrêt par l'API ou l'interface).

Étapes

1. Informez les équipes qui exploitent les alertes : pendant l'arrêt, la sonde n'exécute plus aucun contrôle et peut apparaître `STALE` dans les plateformes de supervision.
2. Si votre plateforme le permet, planifiez une période de maintenance pour éviter les fausses alertes.
3. Déclenchez l'arrêt par l'une des méthodes suivantes :
   - depuis l'interface d'administration, fonction d'arrêt de la sonde ;
   - par l'API : `POST /api/v1/system/shutdown` (réponse `202 Accepted`, l'arrêt suit dans les secondes qui viennent) ;
   - par un appui bref sur le bouton d'alimentation du Raspberry Pi 5, qui déclenche l'arrêt propre (comportement standard de Raspberry Pi OS Lite). Un appui long force la coupure sans arrêt propre : réservez-le aux situations où le système ne répond plus ;
   - en dernier recours, par SSH (si activé) : `sudo systemctl poweroff`.
4. Attendez que le voyant d'état cesse de clignoter et reste allumé en rouge fixe, signe que le système est arrêté (comportement standard du Raspberry Pi 5 : le voyant passe au rouge en veille basse consommation ; l'image de la sonde n'ajoute aucune signalisation propre).
5. Débranchez l'alimentation.

Résultat attendu : la sonde est éteinte et ne répond plus à `GET /health`.

Vérification : au redémarrage suivant, aucun message de récupération de base n'apparaît dans le journal d'audit.

20.2.3 Redémarrer la sonde

Le redémarrage est utile après certaines mises à jour, ou pour rétablir un fonctionnement normal après une saturation de ressources.

1. Déclenchez le redémarrage depuis l'interface d'administration, par l'API (`POST /api/v1/system/reboot`, rôle `admin`) ou, si SSH est activé, par `sudo systemctl reboot`.
2. Attendez le retour de la sonde : `GET /health` répond de nouveau en `200`.
3. Vérifiez dans le tableau de bord que les contrôles reprennent. Après un redémarrage, la sonde recalcule les échéances avec un léger décalage aléatoire, pour ne pas lancer tous les contrôles en même temps. Les exécutions manquées pendant l'arrêt ne sont pas rattrapées.

> **Information** — Une exécution en cours au moment du redémarrage est close avec l'état `UNKNOWN` et le type d'erreur `agent_restart`. Ce n'est pas un incident de la ressource supervisée.

20.2.4 Redémarrages automatiques

La sonde redémarre ses services automatiquement en cas de défaillance logicielle, et l'appareil complet en cas de blocage système (chien de garde matériel). Ces redémarrages sont visibles dans les métriques et le journal d'audit. S'ils se répètent, recherchez une cause matérielle (alimentation, chaleur, carte) avec le chapitre 19.

20.2.5 Coupures électriques

Après une coupure secteur, la sonde redémarre automatiquement dès le retour du courant et reprend ses contrôles sans intervention. Au démarrage, elle vérifie l'intégrité de sa base de données et la restaure si nécessaire à partir de la dernière sauvegarde de configuration (§ 20.5.1).

> ✓ **Conseil** — Branchez la sonde sur une alimentation secourue (onduleur) partagée avec les équipements réseau qui la desservent. C'est la mesure la plus efficace pour préserver la carte microSD, et la sonde continue de superviser pendant les micro-coupures.

20.3 Protection du support de stockage

20.3.1 Comprendre l'usure d'une carte microSD

Une carte microSD utilise de la mémoire flash, dont chaque cellule supporte un nombre limité de cycles d'écriture. Une carte conçue pour un appareil photo n'est pas prévue pour des écritures continues 24 heures sur 24. La sonde réduit volontairement ses écritures :

Mesure intégrée	Effet
Système installé sur une partition en **lecture seule**, données placées sur une partition séparée (ou sur le SSD, § 20.3.4)	Aucune écriture sur la partition système en fonctionnement normal : une coupure n'interrompt aucune écriture du système. Cette protection ne couvre ni une défaillance du support lui-même, ni les écritures légitimes d'une mise à jour sur l'emplacement inactif ; les données restent sur une partition écrite en continu (§ 20.2.5)
Journaux système conservés en mémoire par défaut	Supprime la principale source d'écritures continues
Profil et cache de Chromium en mémoire, détruits après chaque exécution	Aucune écriture disque liée à la navigation
Fichiers temporaires en mémoire	Idem
Regroupement des écritures de résultats	Moins d'opérations d'écriture
Captures d'écran produites seulement en cas d'échec, avec quota	Volume maîtrisé
Purge automatique selon la rétention	Pas de saturation

20.3.2 Recommandations

Recommandation	Justification
Utiliser exclusivement la carte fournie par l'éditeur ; toute carte de remplacement s'obtient selon le § 20.6.4 (une carte achetée ailleurs démarrerait en mode restreint)	La carte fournie, de type « High Endurance », est conçue pour les écritures continues et porte le certificat d'édition

Recommandation	Justification
Pour l'édition Enterprise avec des parcours navigateur fréquents, envisager un SSD NVMe facultatif (carte M.2 HAT+ officielle) comme volume de données. La microSD reste le support principal et suffit à un usage standard	Le SSD reçoit les écritures les plus volumineuses (historique, captures, journaux) et soulage la microSD. Le système continue de démarrer sur la microSD. Activation : § 20.3.4
Protéger l'alimentation (onduleur)	Les coupures sont la première cause de corruption
Ne pas allonger inutilement la rétention	Moins d'écritures et de volume
Ne pas activer durablement les journaux persistants	Réservés aux diagnostics ponctuels
Conserver une carte de rechange préparée (voir § 20.6)	Réduit la durée d'interruption

20.3.3 Durée de vie

La durée de vie d'une carte dépend du nombre de contrôles, de leur fréquence et de la rétention. Elle n'est pas garantie : surveillez les signes d'usure ci-dessous et évaluez chaque année le remplacement préventif de la carte selon l'usage (§ 20.11). Signes d'une carte en fin de vie : erreurs d'entrée/sortie dans les journaux du noyau, redémarrages spontanés, partition de données remontée en lecture seule (la partition système, elle, est en lecture seule par conception), lenteurs inexpliquées (chapitre 19, § 19.3 et § 19.9).

20.3.4 Activer un SSD NVMe

Un SSD NVMe peut compléter la microSD, et dans toutes les éditions. Il est surtout utile pour l'édition Enterprise, dont les scénarios navigateur produisent davantage d'écritures. Il est **facultatif** : la microSD reste le support principal et suffit à un usage standard.

Rôle du SSD. Le SSD sert exclusivement de **volume de données**. La répartition est la suivante :

Support	Contenu
SSD NVMe (s'il est activé)	Historique des exécutions, agrégats, artefacts (captures d'écran), file d'attente d'export, journaux persistants
microSD (toujours)	Partition de démarrage, deux emplacements système (§ 20.4.5), certificat d'édition, configuration et secrets

La sonde démarre toujours sur la microSD : l'activation du SSD ne modifie pas l'ordre de démarrage du Raspberry Pi 5. Le certificat d'édition reste attaché à la carte microSD et le mécanisme de mise à jour est inchangé.

Objectif : déplacer les données volumineuses de la sonde sur un SSD NVMe.

Prérequis

- Matériel monté selon le chapitre 5 : carte M.2 HAT+ officielle de Raspberry Pi, SSD M.2 au format 2230 ou 2242 figurant dans la liste des modèles pris en charge, indiquée dans les notes de version, alimentation officielle de 27 W. Le refroidisseur actif (*Active Cooler*), s'il est utilisé, est installé en premier. Le boîtier officiel n'est utilisable que sans couvercle ni ventilateur.
- Rôle `admin`.
- Export récent de la configuration (§ 20.5.2).
- Une période de maintenance planifiée dans vos plateformes de supervision : la sonde redémarre à la fin de l'opération.

> **Information** — La carte M.2 HAT+ fonctionne en PCIe 2.0 x1 (environ 500 Mo/s). Le mode PCIe Gen 3, non certifié par Raspberry Pi, n'est pas utilisé par la sonde. Ce débit est largement suffisant pour le rôle de volume de données.

> **Avertissement** — Le SSD est **entièrement formaté** lors de l'activation. Toute donnée qu'il contient est définitivement perdue. N'utilisez pas un SSD contenant des données à conserver.

Étapes

1. Arrêtez proprement la sonde (§ 20.2.2), montez la carte M.2 HAT+ et le SSD (chapitre 5), puis remettez la sonde sous tension.
2. Lancez l'activation depuis l'interface d'administration ou par l'API. Le corps de la requête reprend le nom d'hôte de la sonde, ce qui évite d'agir sur la mauvaise sonde :

```bash
# Active le SSD comme volume de données (rôle admin requis).
# Le champ "confirm" doit contenir le nom d'hôte exact de la sonde.
sapi -X POST \
     -H "Content-Type: application/json" \
     -d '{"confirm": "sonde-paris-01"}' \
     "$SONDE/api/v1/system/storage/ssd"
```

3. La sonde formate le SSD, y migre les données concernées (historique, agrégats, artefacts, file d'export, journaux persistants), puis redémarre. Ne coupez pas l'alimentation pendant cette opération.
4. Attendez le retour de la sonde : `GET /health` répond de nouveau en `200`.

Résultat attendu : la sonde fonctionne normalement ; l'historique antérieur est consultable ; les nouvelles données sont écrites sur le SSD.

Vérification : dans la page de santé de l'interface ou dans `GET /api/v1/system`, l'état de santé `storage` est `OK`. Vérifiez que les contrôles ont repris et que l'historique antérieur reste consultable.

Erreurs possibles

Symptôme	Cause probable	Action
Requête refusée	Nom d'hôte incorrect dans `confirm` ; rôle insuffisant	Corriger le nom d'hôte ; utiliser un compte `admin`
SSD non détecté	Nappe PCIe mal insérée ; SSD absent de la liste des modèles pris en charge ; alimentation insuffisante	Arrêter la sonde, vérifier le montage (chapitre 5) et l'alimentation de 27 W
État de santé `storage` en `CRITICAL` après l'activation	SSD défaillant ou déconnecté	Voir le comportement en cas de défaillance ci-dessous

Comportement en cas d'absence ou de défaillance du SSD. Si le SSD activé devient absent ou défaillant, la sonde **continue d'exécuter ses contrôles**. L'historique local est alors conservé sur la microSD, en mode réduit, et l'état de santé `storage` passe en `CRITICAL`. Remplacez le SSD dès que possible ou contactez le support. Surveillez cet état de santé dans vos plateformes (§ 20.7).

> **Attention** — En mode réduit, toutes les écritures se reportent sur la microSD. Ne laissez pas durablement une sonde dans cet état : l'usure de la carte s'accélère.

20.4 Mises à jour

20.4.1 Ce qui est mis à jour

Les mises à jour publiées par l'éditeur couvrent :

- le système d'exploitation (correctifs de sécurité) ;
- le moteur principal et le moteur de scénarios ;
- Chromium et la bibliothèque Puppeteer, toujours livrés **ensemble**, dans des versions testées conjointement par l'éditeur ;
- l'interface d'administration ;
- la liste des plugins Nagios fournis.

Ces éléments ne sont pas mis à jour séparément. Une mise à jour est une **image système complète**, signée par l'éditeur (*bundle*). La sonde vérifie la signature avant toute installation et refuse une image dont la signature est invalide. L'image s'installe sur un second emplacement système, distinct de celui qui fonctionne : en cas d'échec, la sonde revient automatiquement à la version précédente (§ 20.4.5).

Les mises à jour sont publiées sur deux canaux :

Canal	Contenu	Accès
`security`	Correctifs de sécurité, y compris ceux de Chromium	Sans condition, pendant toute la période de support (§ 20.4.6)
`feature` (versions fonctionnelles)	Nouvelles fonctions et évolutions	Réservé aux sondes disposant d'un justificatif de maintenance en cours de validité (§ 20.4.7)

Les deux canaux sont indépendants : leurs images sont distinctes et chacun dispose de son propre mode, `automatic` par défaut pour le canal `security` et `manual` par défaut pour le canal `feature` (§ 20.4.2). Un correctif de sécurité n'exige jamais d'installer une version fonctionnelle.

Une seule image en attente. L'emplacement inactif ne peut contenir qu'**une seule image préparée** à la fois, quel que soit son canal. Les règles suivantes s'appliquent :

Règle	Comportement
Préemption	Une image `security` **préempte** une image `feature` déjà préparée : l'image fonctionnelle est abandonnée, le correctif est installé à sa place, puis la version fonctionnelle est retéléchargée plus tard
Images cumulatives	Une version fonctionnelle contient tous les correctifs de sécurité publiés avant elle
Ligne de versions	Une image `security` s'applique à la ligne de versions installée sur la sonde
Niveau de correctifs	Aucun changement de canal ne peut abaisser le niveau de correctifs déjà installé
Report	Un report vaut pour une version donnée ; une nouvelle image critique est réévaluée, avec un report de 7 jours au plus (§ 20.4.2)

L'image préparée est identifiée dans le champ `staged` de `GET /api/v1/updates` : version, canal et caractère critique.

> **Information** — Chromium n'est jamais mis à jour indépendamment de Puppeteer. Une version de Chromium non validée pourrait modifier le comportement des scénarios. L'éditeur publie les correctifs de sécurité de Chromium après les avoir validés avec la version de Puppeteer embarquée. Il se fixe pour **objectifs**, à compter de la disponibilité du correctif publié par le projet Chromium : 14 jours pour une vulnérabilité critique ou activement exploitée, 30 jours pour une vulnérabilité de gravité élevée. Ces objectifs ne sont pas des engagements contractuels. Les engagements contractuels éventuels relèvent de votre contrat de maintenance.

Le délai qui sépare une vulnérabilité de sa correction effective sur votre sonde comprend plusieurs étapes, qui relèvent de responsabilités distinctes :

Étape	Ce qu'elle recouvre	Responsable
Disponibilité du correctif	Publication de l'image corrigée sur le canal `security` (objectifs ci-dessus, à compter du correctif amont)	Éditeur
Notification	Avis de sécurité publié et joint à la mise à jour ; état de santé `updates` et métrique `sva.update.pending` (§ 20.4.9, § 20.4.11)	Éditeur et sonde
Déploiement	Installation immédiate sur l'emplacement inactif, puis activation selon vos fenêtres, votre préavis et vos éventuels reports (§ 20.4.2, § 20.4.8)	Client
Mesures compensatoires	Dispositions prises en attendant l'activation, par exemple la suspension des scénarios navigateur qui visitent des applications exposées (§ 20.4.11)	Client

Les objectifs de l'éditeur ne portent que sur la première étape. La durée totale dépend aussi de vos fenêtres et de vos reports : plus vos fenêtres sont espacées, plus les mesures compensatoires prennent d'importance.

20.4.2 Choisir un mode de mise à jour

Une mise à jour se déroule en deux temps bien distincts :

Temps	Ce qui se passe	Effet sur la supervision
Installation	Téléchargement de l'image signée, vérification de sa signature et écriture sur l'emplacement système inactif (§ 20.4.5)	Aucun : les contrôles continuent de s'exécuter normalement
Activation	Annonce, avec un préavis qui dépend du cas d'activation (matrice ci-dessous), passage en état de maintenance, drainage des exécutions, redémarrage sur le nouvel emplacement, autotest et, si nécessaire, retour arrière (§ 20.4.9)	Interruption de la supervision assurée par cette sonde, le temps du drainage, du redémarrage et de l'autotest

Un mode par canal. Chacun des deux canaux (§ 20.4.1) dispose de son propre mode :

Canal	Paramètre	Par défaut	Mode `automatic`	Mode `manual`
`security`	`security.mode`	`automatic`	La sonde installe chaque correctif dès qu'il est disponible, sans interrompre les contrôles, puis l'active dans une fenêtre de maintenance, après préavis ; un correctif critique suit `security.criticalPolicy` (§ 20.4.11)	La sonde signale les correctifs disponibles ; un administrateur décide quand les appliquer (§ 20.4.3)
`feature`	`feature.mode`	`manual`	Même fonctionnement pour les versions fonctionnelles auxquelles la sonde a accès (§ 20.4.7)	La sonde signale les versions disponibles ; un administrateur décide quand les appliquer (§ 20.4.3)

Le mode automatique convient aux sondes standard, aux parcs importants et aux équipes qui souhaitent limiter l'exploitation. Le mode manuel convient aux environnements soumis à une gestion des changements stricte et aux sondes pilotes. L'installation automatique des correctifs de sécurité est le réglage par défaut recommandé ; vous pouvez passer le canal `security` en mode `manual` à tout moment, par un simple changement de paramètre.

Fenêtres de maintenance. Une activation automatique ordinaire, ou celle d'un correctif critique en mode `next-window` (réglage par défaut), n'a lieu que dans une **fenêtre de maintenance** définie par vos soins : un créneau récurrent (jours, heure de début, durée) pendant lequel la sonde est autorisée à redémarrer. Vous pouvez définir de 0 à 7 fenêtres. **Aucune fenêtre n'est définie par défaut** et aucune plage horaire n'est imposée. Lors d'une mise en service par l'assistant web, l'étape consacrée aux mises à jour impose un choix explicite : au moins une fenêtre, ou le mode manuel (chapitre 6, § 6.4.5). Dans le fichier `provision.yaml`, la section `updates` est en revanche facultative : en son absence, ou si elle ne contient aucune fenêtre, la sonde n'active aucune mise à jour automatiquement et l'état de santé `updates` reste en `WARNING` jusqu'à la définition d'une fenêtre (chapitre 6, § 6.3.3). La méthode de choix des fenêtres est décrite au § 20.4.8.

Sans fenêtre définie, la sonde n'active d'elle-même ni une mise à jour ordinaire ni un correctif critique en mode `next-window`. La mise à jour reste en attente : l'état de santé `updates` passe en `WARNING`, ou en `CRITICAL` pour un correctif critique, et un administrateur l'active depuis l'interface ou par `POST /api/v1/updates/apply`, immédiatement ou à une heure donnée (§ 20.4.3). Seule exception : un correctif critique en mode `asap`, réglage que vous choisissez explicitement (§ 20.4.11), est activé à l'instant annoncé, même sans fenêtre.

Matrice des activations. Une activation relève toujours de l'un des six cas suivants. Cette matrice fait foi pour l'ensemble de la documentation : les règles de préavis, de fenêtre et de report décrites ailleurs en sont des extraits.

Cas	Déclenchement	Instant de l'activation	Préavis minimal
Manuelle immédiate	Administrateur : `POST /api/v1/updates/apply` sans `scheduledAt`, après confirmation explicite	Aussitôt, dans une fenêtre ou non	Aucun : `update.scheduled` et `maintenance.started` sont émis ensemble
Manuelle programmée	Administrateur : `POST /api/v1/updates/apply` avec `scheduledAt`	`scheduledAt`, dans une fenêtre ou non	1 minute : une date plus proche ou passée est refusée (`422`)
Automatique ordinaire	Sonde, canal en mode `automatic`	Début de la première fenêtre qui commence au moins `notice` après l'annonce	`notice` (24 heures par défaut)
Critique `next-window` (par défaut)	Sonde, canal `security` en mode `automatic`	Début de la première fenêtre qui commence au moins 15 minutes après l'annonce. Une fenêtre plus proche, ou déjà commencée, n'est pas admissible : l'activation attend la fenêtre admissible suivante	15 minutes
Critique `asap` (choix explicite)	Sonde, canal `security` en mode `automatic`	15 minutes après l'annonce, augmentées du décalage `security.criticalStagger`, dans une fenêtre ou non, y compris sans fenêtre définie	15 minutes, augmentées du décalage
Critique, canal en mode `manual`	Administrateur uniquement	Celui que choisit l'administrateur, comme pour une activation manuelle	Celui du cas manuel retenu

Cas	Sans fenêtre définie	Après un report (`postpone`)	Fin prévue (`windowEnd`)
Manuelle immédiate	Possible	Sans objet : l'activation a commencé	`null`

Cas	Sans fenêtre définie	Après un report (`postpone`)	Fin prévue (`windowEnd`)
Manuelle programmée	Possible	Replanifiée selon le mode du canal, comme une activation non programmée	Fin de la fenêtre qui contient `scheduledAt` ; sinon `null`
Automatique ordinaire	Aucune activation : mise à jour en attente	Première fenêtre qui commence après `until` et au moins `notice` après la réannonce ; report de 30 jours au plus	Fin de la fenêtre retenue
Critique `next-window`	Aucune activation automatique : correctif en attente d'un administrateur	Première fenêtre qui commence après `until` et au moins 15 minutes après la réannonce ; report de 7 jours au plus	Fin de la fenêtre retenue
Critique `asap`	Activation à l'instant annoncé	À `until`, ou 15 minutes après la réannonce si `until` est plus proche, sans nouveau décalage ; report de 7 jours au plus	`null`
Critique, canal en mode `manual`	Aucune activation automatique	Sans objet tant qu'aucune activation n'est programmée	Selon le cas manuel retenu

Règles communes à tous les cas :

- **signalement du préavis** : `noticeReduced` vaut `true` dès que le préavis effectif est inférieur à `notice`, dans la réponse de `POST /api/v1/updates/apply`, dans `scheduledActivation` et dans l'attribut `sva.update.notice_reduced` de l'annonce ;
- **réannonce** : toute activation déplacée (report, nouvelle date, activation manuelle qui avance une activation annoncée, changement de fuseau) est annoncée de nouveau, avec le même identifiant d'activation et un numéro d'annonce plus grand. Lorsqu'aucun instant ne peut être déterminé, par exemple après un report sans fenêtre définie, l'activation est annulée (événement `update.cancelled`) et la mise à jour reste en attente (§ 20.4.9) ;
- **état de santé `updates`** : `CRITICAL` tant qu'un correctif critique n'est pas actif, quel que soit le cas ; sinon `WARNING` lorsqu'une mise à jour attend sans activation planifiée, après un report, ou lorsqu'un canal est en mode `automatic` sans aucune fenêtre définie (configuration incomplète), et `OK` dans les autres cas. Pendant l'activation, `/health` répond `MAINTENANCE` (§ 20.4.9) ;
- **notification des plateformes** : chaque annonce, annulation, début et fin de maintenance est transmis aux destinations d'export (événements `update.scheduled`, `update.cancelled`, `maintenance.started`, `maintenance.ended`, métriques du § 20.4.9). Les activations sans fin prévue (`windowEnd` à `null`) demandent une préparation particulière de vos plateformes (§ 20.4.10) ;
- **activation immédiate** : un administrateur peut toujours autoriser explicitement une activation immédiate, y compris pour un correctif critique qui attend une fenêtre admissible.

Préavis. Le préavis `notice` (24 heures par défaut, de `0` à `7d`) s'applique aux activations automatiques ordinaires : si la prochaine fenêtre commence avant son expiration, l'activation passe à la fenêtre suivante. Les correctifs critiques suivent le préavis de 15 minutes de la matrice, et les activations demandées par un administrateur les règles de priorité ci-dessous. Les moyens par lesquels l'activation est annoncée sont décrits au § 20.4.9.

Priorité des activations. Lorsque plusieurs règles peuvent s'appliquer, la sonde les combine ainsi :

Situation	Comportement
Activation automatique ordinaire	Respecte le préavis `notice` ; sinon, elle passe à la fenêtre suivante
Correctif critique (`next-window` ou `asap`)	Respecte le préavis de 15 minutes de la matrice ci-dessus, augmenté du décalage en mode `asap`

Situation	Comportement
`POST /api/v1/updates/apply` **sans** `scheduledAt`	Activation manuelle **immédiate, sans préavis**. L'interface affiche ses conséquences (interruption de la supervision assurée par la sonde, passage en état de maintenance) et demande une confirmation explicite. Les événements `update.scheduled` et `maintenance.started` sont émis aussitôt
`POST /api/v1/updates/apply` **avec** `scheduledAt` à une date passée ou à moins d'une minute	Demande refusée : `422 Unprocessable Content`
`POST /api/v1/updates/apply` avec `scheduledAt` antérieur à l'expiration du préavis	Activation acceptée à la date demandée ; la réponse porte `noticeReduced: true` pour signaler que le préavis habituel n'est pas respecté
Activation manuelle alors qu'un report est en cours	L'activation manuelle **prime** sur le report

> **Attention** — Une intégration qui attend le préavis `notice` avant toute maintenance peut recevoir `update.scheduled` et `maintenance.started` presque simultanément lors d'une activation manuelle immédiate. Planifiez une période de maintenance ponctuelle dans vos plateformes avant de lancer une telle activation (§ 20.4.10).

Report. Un administrateur peut reporter une activation annoncée, par l'interface ou par `POST /api/v1/updates/postpone`, en indiquant la date limite du report dans le champ `until`. L'activation est ensuite replanifiée et réannoncée selon son cas (matrice ci-dessus). Le report est limité à **30 jours** pour un correctif ordinaire et à **7 jours** pour un correctif de sécurité critique (§ 20.4.11). Chaque report est consigné dans le journal d'audit. Un report vaut pour **une version donnée** : si une nouvelle image critique arrive pendant le report, elle est réévaluée et ne peut être reportée que de 7 jours au plus. Une activation manuelle (§ 20.4.3) prime sur le report en cours.

```bash
# Reporte l'activation de la mise à jour en attente (rôle admin requis).
# Date fictive, exprimée en UTC comme toutes les dates de l'API.
sapi -X POST \
     -H "Content-Type: application/json" \
     -d '{"until": "2026-12-01T00:00:00Z"}' \
     "$SONDE/api/v1/updates/postpone"
```

Drainage. Au début de l'activation, la sonde ne lance plus aucune exécution et laisse celles en cours se terminer pendant au plus `drainTimeout` (5 minutes par défaut, 30 minutes au maximum), avant de redémarrer (§ 20.4.9).

Paramètres. La politique se règle dans les paramètres de la sonde, section `updates` (interface d'administration, ou `GET` / `PUT /api/v1/settings/updates`) :

Champ	Description	Valeur par défaut
`security.mode`	Mode du canal `security` : `automatic` ou `manual`	`automatic`
`security.criticalPolicy`	Activation d'un correctif de sécurité critique : `next-window` (au début de la première fenêtre qui commence au moins 15 minutes après l'annonce) ou `asap` (15 minutes après l'annonce, augmentées de `security.criticalStagger`, même hors fenêtre), § 20.4.11	`next-window`
`security.criticalStagger`	Décalage fixe, propre à la sonde, ajouté au préavis réduit d'une activation critique en mode `asap`, de `0` à `4h` ; sert à échelonner les sondes d'un même site (§ 20.4.11)	`0` (aucun décalage)
`feature.mode`	Mode du canal `feature` : `manual` ou `automatic`	`manual`

Champ	Description	Valeur par défaut
`windows`	Liste de 0 à 7 fenêtres de maintenance	Aucune fenêtre
`windows[].days`	Jours de la semaine où la fenêtre s'applique, parmi `mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun`	—
`windows[].start`	Heure de début de la fenêtre, au format `HH:MM`	—
`windows[].duration`	Durée de la fenêtre, au format des autres durées (par exemple `2h` ou `90m`)	—
`notice`	Préavis minimal d'une activation automatique ordinaire, de `0` à `7d` ; les autres cas suivent la matrice ci-dessus	`24h`
`drainTimeout`	Délai accordé aux exécutions en cours pour se terminer au début d'une activation, `30m` au maximum	`5m`
`repositoryUrl`	Adresse du dépôt de mises à jour. Facultatif : par défaut, l'adresse du service de distribution de l'éditeur, `https://updates-synovex.incitypark.fr`, lue dans le profil éditeur. Renseignez-le pour utiliser un miroir interne à votre organisation	Dépôt de l'éditeur

Exemple de section `updates` (valeurs fictives, à adapter à votre organisation selon le § 20.4.8) :

JSON

```
{
  "security": {
    "mode": "automatic",
    "criticalPolicy": "next-window"
  },
  "feature": {
    "mode": "manual"
  },
  "windows": [
    { "days": ["mon", "wed"], "start": "21:30", "duration": "90m" }
  ],
  "notice": "24h",
  "drainTimeout": "5m"
}
```

BASH

```
# Applique la politique de mises à jour contenue dans le fichier updates.json (rôle admin requis).
# 1. Lire la version courante de la section (en-tête ETag)
ETAG=$(sapi -D - -o /dev/null "$SONDE/api/v1/settings/updates" | awk 'tolower($1)=="etag:" {print
    $2}' | tr -d '\r')
# 2. Enregistrer la nouvelle politique (If-Match : version lue à l'étape 1)
sapi -X PUT \
     -H "If-Match: $ETAG" \
     -H "Content-Type: application/json" \
     --data-binary @updates.json \
     "$SONDE/api/v1/settings/updates"
```

Les heures de `windows[].start` sont interprétées dans le fuseau horaire de la section `time` (fuseau choisi à la mise en service, UTC par défaut) ; elles suivent l'heure légale de ce fuseau. Le traitement des changements d'heure, y compris la fin d'une fenêtre traversée par une bascule, est décrit au § 20.4.8. Choisissez une fenêtre suffisamment longue : elle doit couvrir le drainage (`drainTimeout`), le redémarrage, la période de validation de 10 minutes et, le cas échéant, un retour arrière (§ 20.4.5).

Information — Le règlement européen sur la cyberrésilience (CRA) demande que les mises à jour de sécurité automatiques soient, *le cas échéant*, installées dans un délai approprié et activées par défaut, avec un mécanisme de désactivation clair et simple, la notification des mises à jour disponibles et la possibilité de les reporter temporairement. Il demande aussi de séparer les correctifs de sécurité des mises à jour fonctionnelles lorsque c'est techniquement possible. La sonde met en œuvre ces principes par le canal `security` en mode automatique par défaut, désactivable par le paramètre `security.mode`, par l'annonce de chaque activation (§ 20.4.9) et par le report (`POST /api/v1/updates/postpone`). Le règlement n'impose **ni horaire ni calendrier** particulier : le choix des fenêtres relève entièrement de votre organisation.

Conseil — Dans un parc de plusieurs sondes, laissez une ou deux **sondes pilotes** en mode manuel et appliquez-y chaque nouvelle version quelques jours avant les autres. Vérifiez que tous les scénarios restent `OK`, puis généralisez.

Information — Les adresses de l'éditeur utilisées par la sonde (dépôt de mises à jour, documentation en ligne, support) ne sont pas inscrites dans le logiciel. Elles figurent dans un **profil éditeur signé**, livré avec l'image et actualisé par le canal `security`. Un changement d'adresse ne nécessite donc pas de nouvelle version du logiciel. Valeurs du profil : distribution des mises à jour `https://updates-synovex.incitypark.fr`, documentation `https://docs-synovex.incitypark.fr`, support `https://support-synovex.incitypark.fr`, site commercial `https://synovex.incitypark.fr` ; l'adresse de la console centrale prévue, `https://fleet-synovex.incitypark.fr`, y est seulement réservée et n'est jamais contactée par la sonde autonome. Le paramètre `repositoryUrl` permet en outre de pointer vers un miroir interne : comme chaque image est signée par l'éditeur et vérifiée avant installation, un miroir préserve l'authenticité des images ; leur fraîcheur dépend en revanche de la synchronisation du miroir (encadré suivant).

Attention — Avec un miroir, l'**intégrité** et l'authenticité des images restent garanties par leur signature, toujours vérifiée. Leur **fraîcheur** n'est assurée que si les métadonnées du miroir sont à jour : les métadonnées du dépôt sont signées et portent une date d'expiration (`validUntil`, 7 jours). Une sonde refuse des métadonnées expirées, ce qui la protège contre un miroir figé au-delà de cette échéance ; avant celle-ci, un miroir en retard peut encore lui présenter des métadonnées valides qui ne mentionnent pas les correctifs publiés depuis sa dernière synchronisation. Synchronisez donc votre miroir nettement plus souvent que cette échéance, par exemple chaque jour. L'effet de métadonnées expirées sur l'état de santé `updates` est précisé dans les notes de version. Les règles de confiance sont décrites au § 20.4.5.

20.4.3 Appliquer une mise à jour manuellement

Objectif : installer et activer une mise à jour disponible ou en attente, au moment choisi par un administrateur : sonde dont le canal concerné est en mode manuel, sonde sans fenêtre de maintenance, ou activation à avancer (correctif critique, § 20.4.11).

Prérequis : rôle `admin` ; accès HTTPS sortant vers le dépôt de l'éditeur (`https://updates-synovex.incitypark.fr` ou votre miroir interne), ou, à défaut, image signée importée hors ligne (§ 20.4.4) ; sauvegarde récente de la configuration (§ 20.5).

Étapes

1. Consultez les mises à jour disponibles dans l'interface ou par l'API :

   ```bash
   sapi "$SONDE/api/v1/updates"
   ```

   Le champ `scheduledActivation` indique, le cas échéant, l'activation déjà planifiée.
2. Lisez les notes de version (documentation en ligne, ou document transmis par le canal convenu dans votre contrat, annexe L, § 21.12) : changements de comportement, nouvelle version de Chromium et de Puppeteer, actions requises.
3. Exportez la configuration (§ 20.5.2).
4. Planifiez une période de maintenance dans vos plateformes de supervision (§ 20.4.10).
5. Lancez l'activation depuis l'interface ou par l'API, immédiatement :

   ```bash
   sapi -X POST \
        "$SONDE/api/v1/updates/apply"
   ```

   ou à une heure donnée, par le champ facultatif `scheduledAt` (date fictive, exprimée en UTC) :

   ```bash
   sapi -X POST \
        -H "Content-Type: application/json" \
        -d '{"scheduledAt": "2026-11-24T21:00:00Z"}' \
        "$SONDE/api/v1/updates/apply"
   ```

   L'activation **immédiate** (sans `scheduledAt`) a lieu **sans préavis** : dans l'interface, elle n'est lancée qu'après l'affichage de ses conséquences et une confirmation explicite. Ce message rappelle que la sonde cessera de superviser jusqu'à son retour et, sur un site redondant, que la supervision du site sera entièrement interrompue si une autre sonde est déjà en maintenance ou hors service. Le décalage `security.criticalStagger` ne s'applique pas à une activation manuelle (§ 20.4.11). Avec `scheduledAt`, une date passée ou située à moins d'une minute est refusée (`422`) ; une date antérieure à l'expiration du préavis `notice` est acceptée, et la réponse porte alors `noticeReduced: true`. Une activation manuelle prime sur un report en cours (§ 20.4.2).

   > ! **Attention** — Sur un site équipé de plusieurs sondes redondantes, n'activez **jamais manuellement deux sondes du même site en même temps** : attendez la fin de maintenance de la première (événement `maintenance.ended`, retour arrière éventuel compris) avant d'activer la suivante (§ 20.4.12).

   La réponse `202 Accepted` indique que l'opération a démarré. Elle porte l'identifiant de l'activation (`activationId`), l'instant retenu (`at`) et `noticeReduced` ; conservez l'identifiant pour la vérification de l'étape 7. L'en-tête `Location` désigne la ressource de suivi. Si l'image n'est pas encore installée, elle l'est d'abord sur l'emplacement système inactif, pendant que la sonde continue de superviser.
6. Ne coupez pas l'alimentation. Au moment de l'activation, la sonde passe en état de maintenance et draine ses exécutions (§ 20.4.9), redémarre une fois sur le nouvel emplacement, puis vérifie son propre fonctionnement pendant 10 minutes au plus (§ 20.4.5).
7. Une fois la sonde revenue, vérifiez dans `GET /api/v1/updates` que `lastResult.activationId` est l'identifiant renvoyé à l'étape 5, que `lastResult.outcome` vaut `succeeded` et que l'emplacement actif porte la nouvelle version, puis confirmez la version dans `GET /api/v1/system`. Un `lastResult` qui porte un autre identifiant concerne une tentative antérieure (§ 20.4.12).

Résultat attendu : la sonde fonctionne avec la nouvelle version, validée par son autotest, et les contrôles ont repris. Si la validation a échoué, la sonde est revenue automatiquement à la version précédente : l'emplacement actif porte l'ancienne version et l'événement figure dans le journal d'audit.

Vérification : lancez une exécution d'essai des scénarios les plus importants (`POST /api/v1/checks/{id}/runs?dryRun=true`) et comparez leurs durées avec celles observées avant la mise à jour.

En cas de problème : chapitre 19, § 19.10.

20.4.4 Mise à jour hors ligne

Pour une sonde sans accès au dépôt de l'éditeur, un administrateur importe le **même fichier d'image signé** que celui distribué en ligne. Il le télécharge depuis un poste disposant d'un accès Internet, sur le service de distribution de l'éditeur (`https://updates-synovex.incitypark.fr`), ou le reçoit par le canal convenu dans votre contrat tant que ce service n'est pas disponible (annexe L, § 21.12). Il l'importe ensuite par l'interface ou par l'API, puis l'applique comme au § 20.4.3 :

```bash
# Importe un fichier d'image signé téléchargé au préalable (nom fictif).
# TYPE_IMAGE : type de contenu de l'image, indiqué dans les notes de version.
sapi -X POST \
     -H "Content-Type: $TYPE_IMAGE" \
     --data-binary @sva-image-exemple.bundle \
     "$SONDE/api/v1/updates/upload"
```

La sonde vérifie la signature du fichier de la même manière qu'en ligne et refuse un fichier altéré. L'installation, le redémarrage, l'autotest et le retour arrière éventuel se déroulent ensuite exactement comme pour une mise à jour en ligne (§ 20.4.5). Les correctifs du canal `security` peuvent être importés de cette façon pendant toute la période de support, sans justificatif de maintenance.

Un fichier importé hors ligne n'est pas accompagné des métadonnées du dépôt : il ne fait donc l'objet d'**aucun contrôle de fraîcheur**. La règle **anti-retour en arrière** s'applique en revanche : un fichier dont la version est inférieure à la version installée est refusé (§ 20.4.5). Il vous appartient de vérifier, avant l'import, que vous disposez de la dernière image publiée pour votre ligne de versions.

> **Information** — Le nom du fichier ci-dessus est fictif. Utilisez le fichier tel qu'il a été téléchargé, sans le renommer ni le décompresser.

20.4.5 Déroulement d'une mise à jour et retour arrière automatique

La carte microSD contient **deux emplacements système**, A et B. Un seul est actif à un instant donné. Le système y est installé en **lecture seule** : il ne peut pas être modifié en fonctionnement. Les données de la sonde (configuration, résultats, historique) sont stockées sur une partition séparée, ou sur le SSD s'il est activé (§ 20.3.4) : elles sont

communes aux deux emplacements.

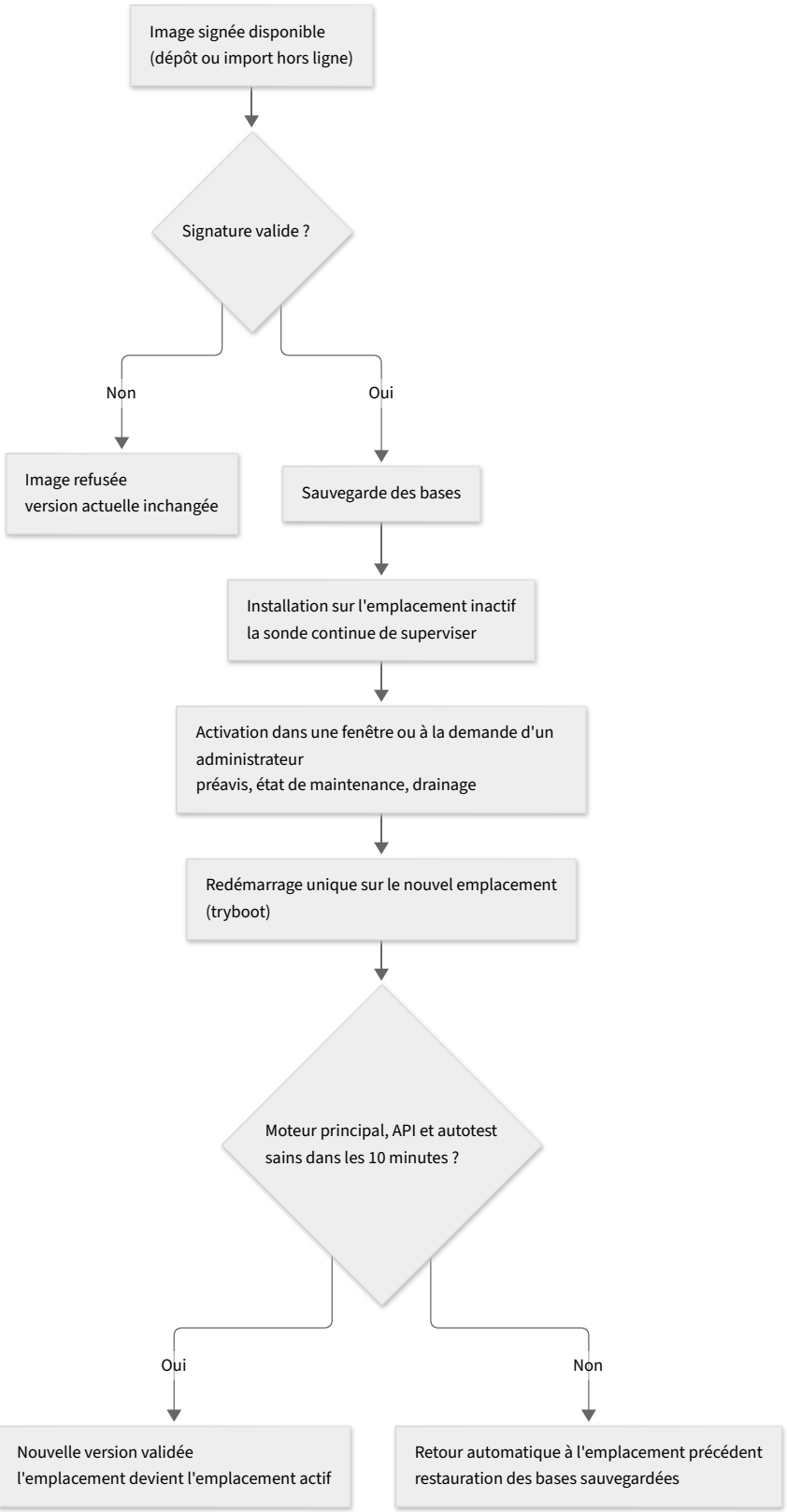


Schéma 20-1 — Déroulement d'une mise à jour et retour arrière automatique

1. **Vérification.** La sonde vérifie la signature de l'image avant toute installation. Pour une image obtenue en ligne, elle vérifie aussi la signature et la date d'expiration des métadonnées du dépôt ; elle refuse une image de version inférieure à la version installée (règles de confiance ci-dessous).

2. **Sauvegarde.** Les bases de données sont sauvegardées avant toute migration de leur structure.
3. **Installation.** L'image est écrite sur l'emplacement inactif, dès qu'elle est disponible pour un canal en mode automatique, ou à la demande d'un administrateur. L'emplacement actif n'est pas modifié et la sonde continue d'exécuter ses contrôles.
4. **Redémarrage d'essai.** Au moment de l'activation, dans une fenêtre de maintenance ou à la demande d'un administrateur, et après le préavis et le drainage des exécutions (§ 20.4.9), la sonde redémarre **une seule fois** sur le nouvel emplacement, au moyen du mécanisme *tryboot* du chargeur d'amorçage du Raspberry Pi : si ce démarrage échoue, le démarrage suivant se fait automatiquement sur l'emplacement précédent. L'indicateur *tryboot* est **à usage unique** : il ne vaut que pour ce démarrage. Le chien de garde du chargeur d'amorçage puis le chien de garde matériel du Raspberry Pi surveillent le démarrage, de sorte qu'un blocage du système avant le démarrage des services provoque une réinitialisation de la carte sans intervention. La surveillance du chargeur d'amorçage (paramètre `BOOT_WATCHDOG_TIMEOUT`) prend fin au démarrage du processeur ARM, d'après la documentation du fabricant ; le chien de garde matériel est ensuite armé par le système. **Toute réinitialisation pendant l'essai** (blocage, coupure électrique, chien de garde) ramène la sonde sur l'emplacement précédent.
5. **Validation.** Le moteur principal, l'API et un autotest doivent être sains dans les **10 minutes**. Dans ce cas, la nouvelle version est validée et devient définitive. Ces 10 minutes sont le délai accordé à l'autotest, et non la durée d'un retour en fonctionnement : si le système se bloque sans que l'autotest puisse conclure, le retour à l'emplacement précédent suppose un redémarrage effectif, provoqué par un chien de garde ou, à défaut, par une remise sous tension ; la durée totale dépend alors de ces mécanismes (étape 4). Le **point de validation définitive** est la réussite de l'autotest : c'est seulement à ce moment que le fichier de configuration du chargeur d'amorçage `autoboot.txt` est mis à jour pour faire du nouvel emplacement l'emplacement de démarrage par défaut. Avant ce point, tout redémarrage ramène à l'emplacement précédent.
6. **Retour arrière.** Dans le cas contraire, la sonde **revient automatiquement** à l'emplacement précédent et restaure les bases sauvegardées. Aucune intervention n'est nécessaire. Comme les données sont communes aux deux emplacements, la restauration des bases sauvegardées à l'étape 2 annule les migrations de structure éventuellement effectuées par la nouvelle version pendant l'essai.

`GET /api/v1/updates` indique la version installée sur chaque emplacement, l'emplacement actif, les mises à jour disponibles, la date de fin de la période de support (`supportEndsAt`) et la date de fin du justificatif de maintenance (`maintenanceEndsAt`). Le champ `staged` identifie l'image préparée sur l'emplacement inactif (version, canal, caractère critique ; § 20.4.1).

Règles de confiance et de fraîcheur des images

Règle	Comportement
Signature des images	Chaque image est signée par l'éditeur ; une signature invalide entraîne le refus de l'image
Métadonnées du dépôt	Signées, avec une date d'expiration (`validUntil`, 7 jours). Des métadonnées expirées sont refusées, ce qui protège contre le gel d'un dépôt ou d'un miroir
Anti-retour en arrière	Une image de version inférieure à la version installée est refusée. Seule exception : le retour arrière A/B local vers l'emplacement précédent (étape 6)
Import hors ligne	Pas de contrôle de fraîcheur (pas de métadonnées du dépôt) ; la règle anti-retour en arrière s'applique (§ 20.4.4)
Clés	La clé racine de confiance est incluse dans l'image. La rotation des clés de signature et la liste de révocation sont distribuées par le canal `security`, dans le profil éditeur signé (§ 20.4.2)
Heure non synchronisée	La date d'expiration des métadonnées ne peut être vérifiée qu'avec une heure fiable (chapitre 7, § 7.4). Le traitement des métadonnées et du profil éditeur tant que l'indicateur `clock` est en `WARNING` est précisé dans les notes de version ; rétablissez d'abord une heure fiable

Règle	Comportement
Sonde hors ligne ou longtemps déconnectée	Une image signée importée hors ligne apporte le profil éditeur de sa version, si la sonde peut la vérifier avec les clés qu'elle détient. Les autres moyens d'actualiser la chaîne de confiance sont précisés dans les notes de version
Métadonnées ou profil expirés, clé révoquée	Refus ; la sonde conserve son profil précédent, n'installe aucune mise à jour en ligne et continue d'exécuter ses contrôles. Pour la récupération, contactez le support (annexe L)

Ces règles s'inspirent des principes de The Update Framework (TUF), cadre public de sécurisation des systèmes de mise à jour.

La durée d'indisponibilité d'une activation (drainage, redémarrage, autotest) et celle d'un retour arrière automatique s'observent sur vos sondes entre les événements `maintenance.started` et `maintenance.ended` (§ 20.4.9).

> **Information** — Le redémarrage interrompt les contrôles exécutés par la sonde. Avant de redémarrer, la sonde draine ses exécutions ; celles qui ne sont pas terminées à l'issue du délai `drainTimeout` sont closes en `UNKNOWN` avec le type d'erreur `maintenance` (§ 20.4.9). Pour éviter les fausses alertes, alignez les périodes de maintenance de vos plateformes sur vos fenêtres (§ 20.4.10) ; pour maintenir la supervision pendant ce redémarrage, voir § 20.4.12.

> **Attention** — Un retour arrière automatique signale un problème de la nouvelle version sur votre sonde. Consultez le journal d'audit, puis contactez le support en joignant une archive de diagnostic (chapitre 19, § 19.12). Ne relancez pas la même mise à jour avant l'avis du support.

20.4.6 Période de support

Chaque version majeure du logiciel bénéficie d'une **période de support** d'au moins **5 ans** à compter de sa mise sur le marché. Cette durée est allongée si la durée d'utilisation prévisible du produit ou la réglementation l'exige : la période retenue est d'**au moins 5 ans, ajustée à la durée d'utilisation prévisible** du produit. Pendant cette période, l'éditeur publie les correctifs de sécurité sur le canal `security`, accessibles à toutes les sondes, avec ou sans contrat de maintenance.

La date de fin de la période de support (mois et année) est indiquée à l'achat. La sonde l'expose dans le champ `supportEndsAt` de `GET /api/v1/system` (et de `GET /api/v1/updates`).

Après cette date, la sonde **continue de fonctionner** avec toutes ses fonctions, mais ne reçoit plus de nouveau correctif. L'état de santé `support` passe alors en `WARNING` pour vous le signaler. Anticipez cette échéance en contactant l'éditeur ou le support (annexe L).

20.4.7 Contrat de maintenance et justificatif

Le contrat de maintenance est **facultatif**. Il donne accès aux **nouvelles versions fonctionnelles** du logiciel. Il n'est pas nécessaire pour recevoir les correctifs de sécurité, qui restent accessibles sans contrat pendant toute la période de support (§ 20.4.6). Ses conditions sont celles de votre contrat de maintenance.

Le contrat se matérialise sur la sonde par un **justificatif de maintenance** : un fichier signé par l'éditeur, distinct du certificat d'édition, qui porte une date de fin. Il s'importe hors ligne, sans connexion à un service de l'éditeur.

Objectif : permettre à la sonde d'installer les versions fonctionnelles couvertes par votre contrat.

Prérequis : rôle `admin` ; fichier de justificatif remis par l'éditeur selon les modalités prévues par votre contrat.

Étapes

1. Importez le justificatif depuis l'interface d'administration ou par l'API :

```bash
# Importe le justificatif de maintenance (nom de fichier fictif).
# TYPE_JUSTIFICATIF : type de contenu du justificatif, indiqué dans les notes de version.
sapi -X PUT \
     -H "Content-Type: $TYPE_JUSTIFICATIF" \
     --data-binary @justificatif-maintenance-exemple \
     "$SONDE/api/v1/system/maintenance"
```

2. Vérifiez la date de fin du justificatif dans le champ `maintenanceEndsAt` de `GET /api/v1/updates`.

Résultat attendu : les versions fonctionnelles couvertes par le contrat apparaissent parmi les mises à jour disponibles.

Erreurs possibles : un justificatif dont la signature est invalide est refusé ; la sonde conserve alors son état antérieur. Vérifiez que vous importez le fichier tel qu'il vous a été remis, sans modification.

> **Information** — L'échéance du justificatif de maintenance n'affecte **aucune fonction acquise** : la sonde continue de fonctionner avec la version installée et de recevoir les correctifs de sécurité jusqu'à la fin de la période de support. Seul l'accès aux nouvelles versions fonctionnelles cesse.

20.4.8 Définir les fenêtres de maintenance et le préavis

Une fenêtre de maintenance est le créneau récurrent pendant lequel la sonde est autorisée à activer une mise à jour, c'est-à-dire à interrompre sa supervision le temps d'un redémarrage. Ce choix vous appartient : la sonde ne propose aucun créneau par défaut, car le meilleur moment dépend de l'activité de votre organisation et des applications supervisées.

Objectif : définir des fenêtres de maintenance et un préavis adaptés à votre organisation, afin que les mises à jour soient activées sans perturber la supervision des périodes importantes.

Prérequis

- Rôle `admin`.
- La liste des plans d'exécution et des contrôles les plus importants de la sonde (chapitre 12).
- Les créneaux de changement de votre organisation (calendrier de gestion des changements, périodes de gel, astreintes).
- Pour un site équipé de plusieurs sondes, l'inventaire des sondes qui exécutent les mêmes contrôles (§ 20.4.12).

Étapes

1. **Identifiez les périodes creuses propres à votre organisation** : les créneaux où les applications supervisées sont le moins utilisées et où une courte interruption de la supervision a le moins de conséquences. Il n'existe pas de créneau universel : une nuit de semaine convient à une application de bureau, mais pas à un site de commerce en ligne dont l'activité culmine le soir, ni à une application qui exécute des traitements par lots la nuit.
2. **Confrontez ces créneaux à vos plans d'exécution** (chapitre 12) : évitez les créneaux où s'exécutent les contrôles les plus importants, par exemple le contrôle d'un traitement par lots ou d'un indicateur métier calculé à heure fixe.
3. **Dimensionnez la durée de chaque fenêtre** : elle doit couvrir le drainage (`drainTimeout`), le redémarrage, la période de validation de 10 minutes et, par précaution, un retour arrière éventuel, qui ajoute un redémarrage. Ces durées s'observent sur vos sondes, par exemple sur une sonde pilote, entre les événements `maintenance.started` et `maintenance.ended` (§ 20.4.5, § 20.4.9).

4. **Choisissez la fréquence** : une ou deux fenêtres par semaine suffisent en général. Davantage de fenêtres réduit le délai entre la publication d'un correctif et son activation ; vous pouvez en définir 7 au plus.
5. **Choisissez le préavis** (`notice`) : il doit laisser à vos équipes le temps de prendre connaissance de l'annonce, de reporter l'activation si nécessaire et de vérifier les périodes de maintenance de leurs plateformes. La valeur par défaut, 24 heures, convient à la plupart des organisations ; la valeur maximale est `7d`. Une valeur de `0` supprime le délai d'annonce : réservez-la aux sondes de test.
6. **Pour plusieurs sondes redondantes d'un même site**, attribuez à chacune des fenêtres qui ne se chevauchent pas, séparées par un écart suffisant pour que la première sonde ait terminé son activation, retour arrière compris, avant que la suivante ne commence (§ 20.4.12). Cet écart doit **dépasser la durée totale mesurée d'une activation** : drainage, redémarrage, autotest et retour arrière éventuel (§ 20.4.5). Des fenêtres disjointes ne suffisent pas si une activation commencée en fin de fenêtre déborde sur la fenêtre de l'autre sonde.
7. **Enregistrez la politique** dans l'interface d'administration ou par `PUT /api/v1/settings/updates` (exemple au § 20.4.2).
8. **Alignez les périodes de maintenance de vos plateformes de supervision** sur ces fenêtres (§ 20.4.10), puis consignez les fenêtres retenues dans votre référentiel de gestion des changements.

Résultat attendu : la section `updates` de la sonde contient vos fenêtres et votre préavis ; l'état de santé `updates` n'est plus en `WARNING` faute de fenêtre.

Vérification

- `GET /api/v1/settings/updates` renvoie les fenêtres et le préavis enregistrés.
- La page de santé de l'interface ou `GET /api/v1/system` n'indique plus d'avertissement lié à l'absence de fenêtre pour l'état de santé `updates`.
- À la prochaine mise à jour ordinaire, le champ `scheduledActivation` de `GET /api/v1/updates` indique une activation située dans l'une de vos fenêtres, annoncée au moins `notice` à l'avance (15 minutes au moins pour un correctif critique).

Erreurs possibles

Situation	Cause probable	Action
Paramètres refusés	Heure de début hors du format `HH:MM` ; plus de 7 fenêtres ; préavis supérieur à `7d` ; `drainTimeout` supérieur à `30m`	Corriger la valeur. Dans `provision.yaml`, écrire l'heure entre guillemets (chapitre 6)
Activation repoussée à une fenêtre ultérieure	La prochaine fenêtre commençait avant l'expiration du préavis : l'activation passe alors à la fenêtre suivante	Comportement normal. Ajouter une fenêtre ou réduire le préavis si ce délai est trop long
Deux sondes du même site redémarrent en même temps	Fenêtres qui se chevauchent ; écart entre fenêtres inférieur à la durée totale d'une activation ; activations manuelles simultanées	Décaler les fenêtres (étape 6) ; activer manuellement les sondes l'une après l'autre (§ 20.4.3)
Fausses alertes pendant la fenêtre	Périodes de maintenance des plateformes non alignées sur les fenêtres	§ 20.4.10 et chapitre 18

Précautions

- Une fenêtre trop courte peut ne pas laisser le temps d'un retour arrière ; préférez une marge confortable.
- Après un changement d'heure légale ou de fuseau horaire, vérifiez que vos fenêtres et les périodes de maintenance de vos plateformes restent alignées.
- Lors des changements d'heure légale, une fenêtre qui commence dans l'heure **inexistante** (passage à l'heure d'été) démarre à la première minute valide qui suit ; une fenêtre qui commence dans l'heure **répétée** (passage à l'heure d'hiver) n'a lieu qu'une seule fois, à la **première occurrence** de cette heure (heure d'été). Configurez les

outils de mise en sourdine de vos plateformes **dans le même fuseau** que la sonde : c'est nécessaire pour éviter un décalage d'une heure à chaque bascule, mais cela ne garantit pas que l'outil résout une heure inexistante ou répétée comme la sonde, chacun appliquant ses propres règles. Vérifiez la correspondance lors de la première bascule, ou évitez de placer le début d'une fenêtre entre 2 h et 3 h du matin.

- Lorsqu'une bascule intervient pendant une fenêtre, sa fin dépend de la sémantique de `windows[].duration` : la durée est un temps écoulé, compté depuis le début effectif de la fenêtre, de sorte que la fenêtre couvre toujours la durée choisie ; sa fin, exprimée en heure locale, se décale alors d'une heure par rapport au calcul « début plus durée » en heure civile. Un outil de sourdine défini par deux bornes civiles peut alors différer d'une heure de la fenêtre de la sonde cette nuit-là.
- Une modification du fuseau de la section `time` recalcule et réannonce les activations planifiées dans une fenêtre, dont les bornes sont des heures locales. Si le préavis devient insuffisant au regard de la nouvelle heure, l'activation passe à la fenêtre suivante. Une activation manuelle programmée par `scheduledAt` désigne un instant absolu en UTC : elle reste fixée à cet instant (chapitre 7, § 7.4).
- Revoyez les fenêtres lorsque l'activité supervisée change (nouvelle application, nouveaux horaires d'ouverture, nouveaux traitements par lots).

> ✓ **Conseil** — Pour un site équipé de deux sondes, une disposition simple consiste à placer leurs fenêtres le même jour de faible activité, à plusieurs heures d'intervalle, ou à des jours différents. Pendant les activations automatiques, la supervision du site reste ainsi assurée par au moins une sonde, à condition que l'écart dépasse la durée totale mesurée d'une activation ; les activations manuelles et `asap` ne sont pas coordonnées par ces fenêtres (§ 20.4.12).

20.4.9 Notification, drainage et état de maintenance

Une activation suit toujours la même chronologie, qu'elle ait lieu dans une fenêtre de maintenance ou à la demande d'un administrateur.

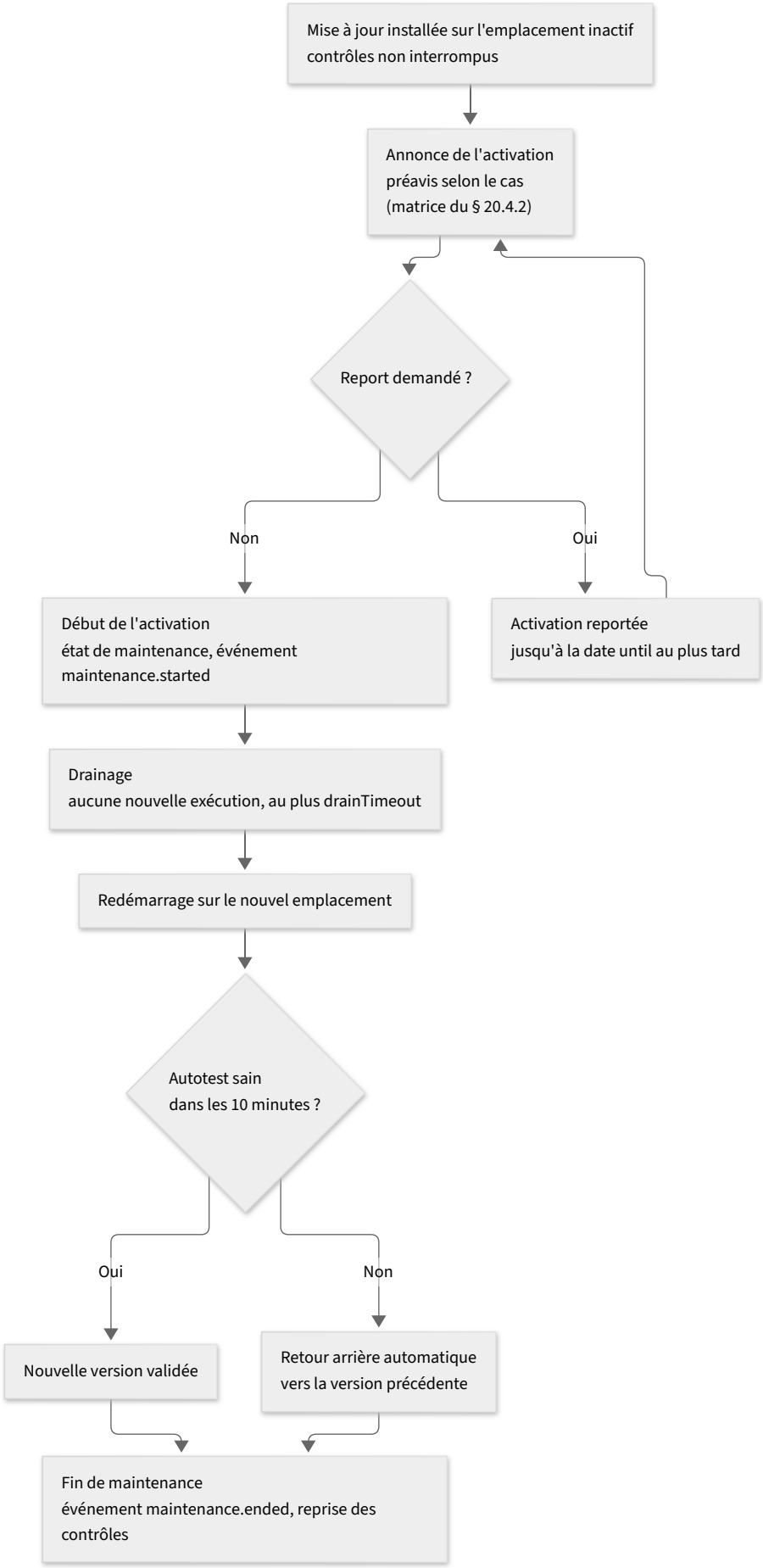


Schéma 20-2 — Chronologie d'une activation : notification, drainage, maintenance

1. Annonce. Dès qu'une activation est planifiée ou replanifiée, la sonde l'annonce par les moyens suivants, avec le préavis que fixe la matrice du § 20.4.2 pour son cas : au moins `notice` pour une activation automatique ordinaire, 15 minutes pour un correctif critique (augmentées du décalage en mode `asap`), le délai choisi pour une activation manuelle programmée, aucun pour une activation manuelle immédiate. Un préavis effectif inférieur à `notice` est signalé par `noticeReduced: true` :

Moyen	Contenu
Interface d'administration	Bandeau indiquant la mise à jour et l'heure prévue de son activation
`GET /api/v1/updates`	Champ `scheduledActivation` : activation planifiée
Journal d'audit	Planification de l'activation, puis report éventuel
Événement OTLP `update.scheduled`	Transmis aux destinations d'export (chapitre 18), à chaque planification et à chaque replanification
Événement OTLP `update.cancelled`	Activation annoncée qui n'aura pas lieu sous cet identifiant : report sans instant déterminable, image remplacée par une autre, canal passé en mode `manual` ou fenêtres supprimées, image retirée
Métrique `sva.update.pending` (unité `{state}`, attributs `sva.update.channel` et `sva.update.critical`)	Mise à jour en attente d'activation, avec son canal et son caractère critique
Métriques `sva.maintenance.next_start` et `sva.maintenance.next_end` (unité `s`, horodatages)	Début et fin **prévue** de la prochaine maintenance planifiée

Ces signaux permettent à vos équipes et à vos plateformes d'anticiper l'activation : vérifier les périodes de maintenance, prévenir les équipes concernées ou reporter l'activation (§ 20.4.2).

Identification et ordre des annonces. Chaque activation planifiée reçoit un **identifiant d'activation** unique (`activationId` dans l'API, attribut `sva.update.activation.id` des événements). Il reste le même tant que la même image doit être activée : report, nouvelle date, activation manuelle qui avance l'activation annoncée, recalcul après un changement de fuseau. Une nouvelle image reçoit un nouvel identifiant, et l'activation de l'image remplacée est annulée par `update.cancelled`. Chaque annonce porte en outre un **numéro d'annonce** (`sva.update.announcement.sequence`, repris par `scheduledActivation.sequence`), strictement croissant pour une même sonde, toutes activations confondues. Ce numéro ne régresse ni après un redémarrage, ni après un retour arrière, ni après une restauration de configuration. Il est dérivé de l'heure de la sonde : après le remplacement d'une sonde qui reprend l'identité de la précédente (§ 20.6), sa croissance n'est assurée que si l'heure de la nouvelle sonde est fiable (indicateur `clock` à `OK`, chapitre 7, § 7.4).

Pour un automatisme qui reçoit ces événements, la règle est simple : pour une sonde donnée, **seule compte l'annonce de plus grand numéro**. Une annonce reçue en double, ou après une annonce de numéro supérieur, est ignorée. Le chapitre 18 (§ 18.15.4) en donne une mise en œuvre pour Alertmanager.

Les attributs de ces quatre événements (`update.scheduled`, `update.cancelled`, `maintenance.started`, `maintenance.ended`) sont décrits au chapitre 18 (§ 18.3.6), qui fait référence : identifiant d'activation, numéro d'annonce, cas d'activation, canal, caractère critique, version, préavis réduit, bornes de maintenance, motif d'annulation et résultat.

Valeurs selon le mode d'activation. L'instant de l'annonce et la borne de fin dépendent du mode d'activation. Dans le tableau suivant, l'**annonce** est l'événement `update.scheduled` ; le **début annoncé** est la valeur de `scheduledActivation.at` et de `sva.maintenance.next_start` ; la **fin prévue** est celle de `windowEnd` et de `sva.maintenance.next_end`.

Mode d'activation	Annonce	`noticeReduced`	Début annoncé	Fin prévue
Automatique ordinaire	Au moins `notice` avant le début de la fenêtre retenue	`false`	Début de la fenêtre retenue	Fin de cette fenêtre
Manuelle immédiate (`POST /api/v1/updates/apply` sans `scheduledAt`)	Au début de l'activation, avec `maintenance.started`	`true`	Heure de réception de la demande (champ `at` de la réponse)	Aucune fenêtre : `windowEnd` vaut `null`
Manuelle à date fixe (`scheduledAt`)	À l'acceptation de la demande	`true` si `scheduledAt` précède l'expiration du préavis `notice` , `false` sinon	`scheduledAt`	Hors fenêtre : `windowEnd` vaut `null` . Date située dans une fenêtre : fin de cette fenêtre
Correctif critique, `criticalPolicy: asap`	Dès l'installation, au moins 15 minutes avant l'activation (§ 20.4.11)	`true` si 15 minutes augmentées du décalage restent inférieures à `notice`	Instant annoncé : 15 minutes après l'annonce, augmentées du décalage `security.criticalStagger` de la sonde (§ 20.4.11)	Aucune fenêtre : `windowEnd` vaut `null`
Correctif critique, `criticalPolicy: next-window`	Dès l'installation, au moins 15 minutes avant le début de la fenêtre retenue	`true` si le délai jusqu'à la fenêtre retenue est inférieur à `notice`	Début de la première fenêtre qui commence au moins 15 minutes après l'annonce	Fin de cette fenêtre

Lorsque `windowEnd` vaut `null` , la métrique `sva.maintenance.next_end` vaut `0` alors que `sva.maintenance.next_start` porte le début annoncé, et l'attribut `sva.maintenance.end` est **absent** de `update.scheduled` : aucune fin n'est annoncée, et la durée de la maintenance se prépare comme indiqué au paragraphe suivant. Sans fenêtre ni activation planifiée, les deux métriques valent `0` .

La **fin prévue** est une borne de planification. La **reprise réelle** est signalée par l'événement `maintenance.ended` (autotest réussi ou retour arrière terminé) : elle peut précéder la fin prévue ou la dépasser, par exemple lorsqu'une activation commencée en fin de fenêtre déborde (§ 20.4.8). Pour une activation hors fenêtre, dimensionnez la période de maintenance de vos plateformes sur une **durée maximale** que vous fixez d'après la durée totale observée d'une activation sur vos sondes, retour arrière compris (§ 20.4.5), puis levez-la dès la reprise vérifiée de la sonde (§ 20.4.12).

2. Début de l'activation et drainage. Au début de l'activation, la sonde passe en **état de maintenance** et émet l'événement `maintenance.started` . Aucune exécution nouvelle ne démarre. Les exécutions en cours disposent du délai `drainTimeout` (5 minutes par défaut, 30 minutes au maximum) pour se terminer normalement. Au-delà, elles sont interrompues et closes avec l'état `UNKNOWN` et le type d'erreur `maintenance` .

3. Redémarrage et autotest. La sonde redémarre sur le nouvel emplacement, puis vérifie son fonctionnement pendant 10 minutes au plus. En cas d'échec, elle revient automatiquement à la version précédente (§ 20.4.5).

4. Fin de maintenance. La sonde émet l'événement `maintenance.ended` , quitte l'état de maintenance et reprend ses contrôles à leur prochaine échéance. Cet événement est émis lorsque l'autotest a réussi, ou lorsque le retour arrière est terminé ; son attribut `sva.update.result` indique lequel. Il signale la fin de la maintenance, pas à lui seul le bon fonctionnement des contrôles : la reprise se vérifie comme au § 20.4.12.

Exposition de l'état de maintenance. Tant que la sonde répond, son état de maintenance est visible par les moyens suivants :

Moyen	Valeur pendant la maintenance
`GET /health`	`{"status": "MAINTENANCE"}`, avec le code HTTP `503`
Métrique `sva.agent.maintenance` (unité `{state}`, Prometheus `sva_agent_maintenance`)	`1`
Événements OTLP	`maintenance.started` au début, `maintenance.ended` à la fin
Plugin `check_sva` (Nagios, Centreon)	Code de retour `3` (`UNKNOWN`), avec le texte `MAINTENANCE` (chapitre 18, § 18.10)

> **Attention** — Pendant le redémarrage proprement dit, la sonde ne répond plus du tout : ni `/health`, ni les métriques, ni `check_sva` ne sont disponibles. L'état de maintenance ne peut donc pas, à lui seul, éviter les alertes de vos plateformes (§ 20.4.10).

Conséquences pour les contrôles

Situation	Conséquence
Exécution en cours au début de l'activation	Se termine normalement si elle s'achève dans le délai `drainTimeout` ; sinon, close en `UNKNOWN` avec le type d'erreur `maintenance`
Exécution prévue pendant la maintenance	Non lancée ; elle n'est pas rattrapée après la reprise (§ 20.2.3)
État `STALE`	Son calcul est suspendu pendant la maintenance et pendant un intervalle de planification de chaque contrôle après la reprise : une maintenance ne rend aucun contrôle `STALE`

> **Information** — Le type d'erreur `maintenance` signale une exécution interrompue par la sonde elle-même pour activer une mise à jour. Comme `agent_restart` (§ 20.2.3), il ne traduit aucun incident de la ressource supervisée. Vous pouvez l'exclure de vos calculs de disponibilité, selon la méthode convenue avec vos parties prenantes (chapitre 17).

20.4.10 Éviter les fausses alertes pendant une maintenance

L'état de maintenance (§ 20.4.9) informe vos plateformes que la sonde va redémarrer, mais il **ne suffit pas à supprimer les alertes**. Pendant le redémarrage, la sonde est injoignable : Prometheus constate l'échec de la collecte, les métriques `sva.agent.up` cessent d'arriver, et `check_sva` ne reçoit aucune réponse. Sans précaution, vos plateformes déclenchent alors les alertes « sonde injoignable » ou « sonde muette », alors qu'il s'agit d'une opération planifiée.

Objectif : éviter que l'activation d'une mise à jour ne déclenche de fausses alertes, sans masquer une véritable panne.

Prérequis : fenêtres de maintenance définies (§ 20.4.8) ; accès à la configuration des alertes de vos plateformes.

Étapes

1. Dans chaque plateforme qui reçoit les données de la sonde, configurez une période de maintenance récurrente **alignée sur chacune des fenêtres** de la sonde (mêmes jours, même heure de début, même durée, même fuseau horaire). Les mécanismes à utiliser sont décrits au chapitre 18 (§ 18.15) : intervalles de temps d'Alertmanager, règles excluant les sondes en maintenance (`sva_agent_maintenance == 1`), temps d'arrêt programmés

récurrents de Datadog, fenêtres de maintenance de Dynatrace, temps d'arrêt récurrents de Centreon, temps d'arrêt programmés de Nagios.

2. Limitez cette période aux alertes qui concernent **la sonde en maintenance** et les contrôles qu'elle exécute. Ne neutralisez pas les alertes des autres sondes du site, qui continuent de superviser (§ 20.4.12).
3. Pour une activation hors fenêtre, c'est-à-dire une activation manuelle (§ 20.4.3) ou un correctif critique en mode `asap` (§ 20.4.11), planifiez une période de maintenance ponctuelle limitée à la sonde concernée : **avant** de lancer une activation manuelle immédiate, car son annonce arrive au moment où elle commence ; **dès l'annonce** dans les autres cas, le préavis étant alors de 15 minutes au moins. Fixez sa durée maximale d'après la durée totale observée d'une activation, retour arrière compris, et levez-la après la reprise vérifiée de la sonde (§ 20.4.12). Une automatisation par événement est décrite au chapitre 18 (§ 18.15.4).
4. Conservez une alerte qui se déclenche si la sonde n'est pas revenue **après la fin de la fenêtre** ou, sans fenêtre, après la durée maximale retenue (§ 20.7.1, alerte « maintenance anormalement longue ») : c'est elle qui révèle une activation qui a mal tourné.

Résultat attendu : une activation dans une fenêtre, ou couverte par une période ponctuelle, ne produit aucune alerte ; une sonde qui ne revient pas après la fin de la fenêtre ou de la période déclenche une alerte.

Vérification : lors de la première activation qui suit la configuration, contrôlez dans vos plateformes qu'aucune alerte n'a été déclenchée pendant la fenêtre, et que les événements `maintenance.started` et `maintenance.ended` ont bien été reçus.

Erreurs possibles

Situation	Cause probable	Action
Alerte « sonde injoignable » pendant la fenêtre	Période de maintenance de la plateforme absente, décalée ou exprimée dans un autre fuseau horaire	Réaligner la période sur la fenêtre de la sonde
Aucune alerte alors que la sonde n'est jamais revenue	Période de maintenance plus longue que la fenêtre, ou alerte de maintenance anormalement longue absente	Limiter la période à la fenêtre ; ajouter l'alerte du § 20.7.1

> **Avertissement** — Une période de maintenance trop large masque les véritables pannes. Ne neutralisez jamais durablement les alertes d'une sonde au motif qu'elle se met à jour.

Les configurations détaillées propres à chaque plateforme figurent au chapitre 18 (§ 18.15.2 à § 18.15.5).

20.4.11 Correctifs de sécurité critiques

Un correctif est dit **critique** lorsqu'il corrige une vulnérabilité activement exploitée. Il suit un traitement accéléré :

1. **Installation immédiate.** Dès sa publication, le correctif est téléchargé et installé sur l'emplacement inactif, sans interrompre les contrôles. Lorsque le canal `security` est en mode `manual`, le correctif n'est jamais activé automatiquement : l'état de santé `updates` passe en `CRITICAL`, la notification est émise, et l'administrateur décide de l'activation.
2. **Avis de sécurité.** L'éditeur publie un avis de sécurité qui décrit la vulnérabilité et le correctif sur le site de l'éditeur ; l'avis est aussi joint à la mise à jour dans `GET /api/v1/updates`.
3. **Signalement sur la sonde.** L'état de santé `updates` passe en `CRITICAL` et la métrique `sva.update.pending` porte l'attribut `sva.update.critical` égal à `true`.
4. **Activation** selon le paramètre `security.criticalPolicy` :

Valeur	Activation	Préavis	Recommandé pour
`next-window` (par défaut)	Au début de la première fenêtre de maintenance qui commence au moins 15 minutes après l'annonce ; une fenêtre plus proche, ou déjà commencée, n'est pas admissible et l'activation attend la suivante	15 minutes au moins	La plupart des organisations : l'interruption reste dans un créneau maîtrisé
`asap`	15 minutes après l'annonce, augmentées du décalage `security.criticalStagger` éventuel de la sonde, sans attendre une fenêtre, y compris si aucune fenêtre n'est définie	15 minutes, augmentées du décalage	Sondes exposées à un risque élevé, ou organisations dont les fenêtres sont espacées

Dans tous les cas, l'activation est annoncée et précédée du drainage des exécutions (§ 20.4.9). **Aucun redémarrage immédiat n'est imposé, et aucun redémarrage hors de vos fenêtres n'a lieu par défaut** : le mode `asap` résulte toujours d'un choix explicite d'un administrateur. Sans fenêtre définie, un correctif critique en mode `next-window` reste en attente, avec l'état de santé `updates` en `CRITICAL`, jusqu'à ce qu'un administrateur l'active (§ 20.4.3) ; en mode `asap`, il est activé à l'instant annoncé. L'activation peut être reportée de **7 jours au plus** (§ 20.4.2), et un administrateur peut toujours l'avancer en l'activant lui-même.

> **Information** — Avec `criticalPolicy: asap`, l'activation peut avoir lieu **hors de vos fenêtres**. C'est, avec l'activation manuelle (§ 20.4.3), l'une des deux exceptions documentées à la règle selon laquelle une activation commence dans l'une des fenêtres que vous avez choisies (§ 20.4.12). Si vous retenez `asap`, prévoyez dans vos plateformes une neutralisation des alertes adaptée à une maintenance non planifiée (§ 20.4.10), sans masquer les alertes des autres sondes du site.

Échelonner les sondes d'un même site (`security.criticalStagger`). Un correctif critique est publié pour toutes les sondes en même temps. En mode `asap`, sans précaution, les sondes d'un même site l'activeraient donc à quelques instants d'intervalle, et la supervision du site serait interrompue. Pour l'éviter, le paramètre `security.criticalStagger` retarde l'activation d'une durée **fixe, propre à chaque sonde** :

- l'activation a lieu 15 minutes après l'annonce, augmentées de ce décalage ; l'instant obtenu est annoncé dès l'installation du correctif, comme toute activation (§ 20.4.9) ;
- la valeur est **déterministe** : elle est choisie par l'administrateur, sans tirage aléatoire, et donne toujours le même ordre d'activation ;
- elle va de `0` (valeur par défaut, aucun décalage) à `4h`. Les métadonnées signées d'un correctif peuvent imposer un plafond inférieur lorsque son urgence l'exige ;
- le décalage est calculé **une seule fois** pour une image préparée. Il ne se renouvelle pas et ne se cumule pas avec un report : un correctif critique n'est jamais retardé indéfiniment, et son report reste limité à 7 jours (§ 20.4.2) ;
- il ne s'applique ni au mode `next-window`, ni au mode `manual`, ni à une activation manuelle.

Sur un site équipé de deux sondes, laissez par exemple `0` sur la première et attribuez à la seconde un décalage supérieur à la durée totale mesurée d'une activation, retour arrière compris (§ 20.4.8), telle que vous l'observez entre les événements `maintenance.started` et `maintenance.ended` (§ 20.4.9).

> **Attention** — Un décalage temporel **ne garantit pas** que l'autre sonde soit opérationnelle. Les sondes sont autonomes : aucune ne connaît l'état des autres. Le décalage ne vérifie ni que la première sonde a terminé son activation, ni qu'elle n'est pas en retour arrière, ni qu'elle n'était pas déjà hors service. Deux activations peuvent encore se recouvrir si l'écart entre les décalages est trop court, ou en cas d'activation manuelle. La sonde ne garantit donc pas l'absence de redémarrage simultané. Pour les environnements critiques, appliquez les recommandations du § 20.4.12.

Objectif : traiter un correctif critique dans les meilleurs délais, en maîtrisant l'interruption de la supervision.

Prérequis : rôle `admin` ; alerte « correctif critique en attente » configurée (§ 20.7.1).

Étapes

1. À réception de l'alerte ou de l'avis de sécurité, consultez `GET /api/v1/updates` : version du correctif, champ `scheduledActivation`.
2. Lisez l'avis de sécurité et évaluez l'exposition de la sonde.
3. Choisissez l'une des options suivantes :
   - laisser l'activation se faire à l'instant annoncé ;
   - l'avancer en l'activant vous-même, immédiatement ou à une heure donnée (§ 20.4.3), après avoir planifié une période de maintenance dans vos plateformes (§ 20.4.10) ;
   - la reporter, pour une raison justifiée (période critique pour l'activité), de 7 jours au plus (`POST /api/v1/updates/postpone`).

   Tant que le correctif n'est pas actif, appliquez si nécessaire des **mesures compensatoires**, par exemple suspendre les scénarios navigateur qui visitent des applications exposées à Internet ou des contenus que vous ne maîtrisez pas, ou restreindre temporairement les flux sortants de la sonde (chapitre 7). Levez-les une fois le correctif actif.
4. Après l'activation, vérifiez la version active dans `GET /api/v1/updates` et le retour de l'état de santé `updates` à `OK`.

Résultat attendu : le correctif est actif ; l'état de santé `updates` n'est plus en `CRITICAL`.

Erreurs possibles : un report au-delà de 7 jours est refusé ; un retour arrière automatique laisse le correctif inactif et doit être signalé au support (§ 20.4.5).

Cas urgents. Lorsque l'avis de sécurité signale une exploitation active susceptible de toucher la sonde elle-même, et que votre prochaine fenêtre est lointaine :

1. Appliquez immédiatement les mesures compensatoires décrites à l'étape 3.
2. Planifiez une période de maintenance ponctuelle dans vos plateformes (§ 20.4.10).
3. Activez le correctif manuellement, immédiatement ou à une heure proche (§ 20.4.3). Sur un site équipé de plusieurs sondes, traitez-les l'une après l'autre, jamais simultanément (§ 20.4.12).
4. Vérifiez la version active et l'état de santé `updates`, puis levez les mesures compensatoires.

Pour un traitement durable, envisagez `criticalPolicy: asap` ou des fenêtres plus rapprochées.

> **Avertissement** — Une sonde qui exécute des scénarios navigateur visite des applications web avec Chromium. Un correctif critique de Chromium réduit un risque réel : ne reportez son activation que pour une raison précise, et jamais par défaut.

20.4.12 Continuité de service et redondance

Une sonde unique **ne supervise pas pendant son propre redémarrage**. Pendant l'activation d'une mise à jour (§ 20.4.9), les contrôles qu'elle exécute sont suspendus : une panne survenant à ce moment ne serait détectée qu'après son retour. Pour la plupart des usages, cette interruption brève et planifiée est acceptable. Pour les environnements critiques, elle ne l'est pas.

Architecture recommandée pour les environnements critiques

Principe	Mise en œuvre
Au moins **deux sondes par site**	Chaque sonde exécute les mêmes contrôles, sur les mêmes cibles

Principe	Mise en œuvre
Fenêtres **qui ne se chevauchent pas**	L'écart entre les fenêtres des sondes d'un même site dépasse la durée totale mesurée d'une activation, retour arrière compris (§ 20.4.8), afin que leurs activations automatiques ne se recouvrent pas ; elles ne sont jamais activées manuellement en même temps (§ 20.4.3). Cet espacement ne coordonne pas les activations `asap`, qui ont lieu hors fenêtre (§ 20.4.11)
Activations critiques **échelonnées**	Avec `criticalPolicy: asap`, un décalage `security.criticalStagger` différent sur chaque sonde du site (§ 20.4.11). Le décalage réduit le risque d'activations simultanées sans le supprimer : il ne vérifie pas l'état des autres sondes
Retour en service **vérifié**	Pour les environnements critiques : fenêtres distinctes avec `criticalPolicy: next-window`, ou orchestration externe qui n'active une sonde qu'après le retour vérifié de la précédente (ci-dessous)
Alertes fondées sur **l'ensemble des sondes** du site	Une alerte ne dépend jamais d'une seule sonde : elle est évaluée à partir des résultats de toutes les sondes qui répondent
Indépendance matérielle	Si possible, alimentations et raccordements réseau distincts (commutateurs différents), pour qu'un même incident local n'affecte pas toutes les sondes

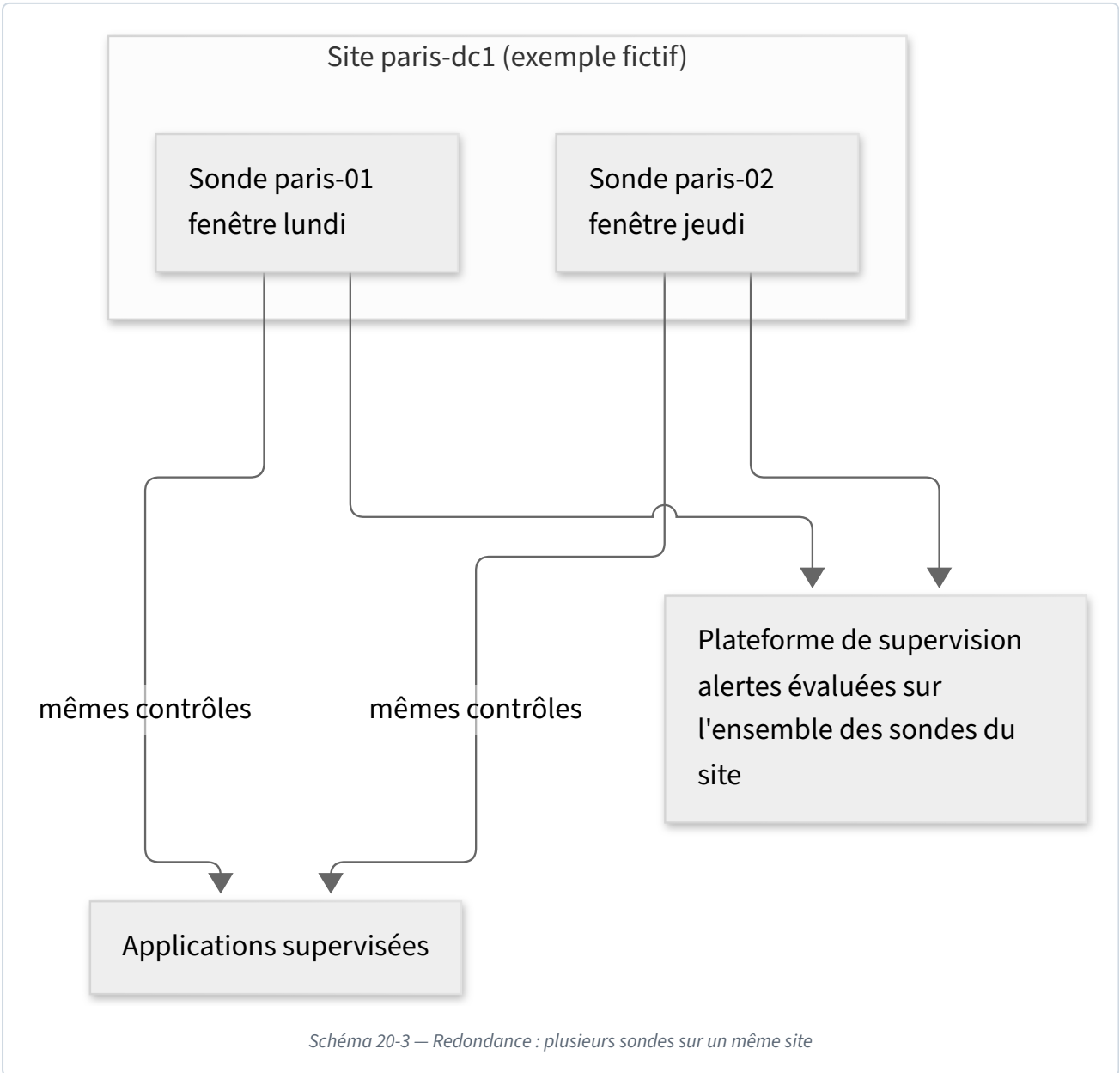


Schéma 20-3 — Redondance : plusieurs sondes sur un même site

Objectif : maintenir la supervision d'un site pendant la maintenance de l'une de ses sondes.

Prérequis

- Au moins deux sondes installées sur le site (chapitres 5 et 6), chacune avec sa propre carte microSD de l'édition requise, et le même paramètre `site`.
- Une plateforme de supervision capable d'agréger les résultats de plusieurs sondes (chapitre 18).

Étapes

1. Installez et mettez en service la seconde sonde, avec le même paramètre `site` que la première (chapitre 6).
2. Exportez la configuration de la première sonde (§ 20.5.2), puis importez-la sur la seconde avec le paramètre `newIdentity=true` (§ 20.5.3) : les deux sondes exécutent les mêmes contrôles, avec les mêmes identifiants de contrôle, tout en restant distinctes dans vos plateformes.
3. Attribuez à chaque sonde des fenêtres de maintenance qui ne se chevauchent pas (§ 20.4.8). L'import ayant recopié la section `updates`, modifiez les fenêtres de la seconde sonde.
4. Créez des règles d'alerte fondées sur l'ensemble des sondes du site (règles d'alerte de site ci-dessous ; groupe `sondes-site` du chapitre 18, § 18.15.6), en complément des alertes de santé propres à chaque sonde (§ 20.7.1).
5. Configurez les périodes de maintenance de vos plateformes sonde par sonde (§ 20.4.10).
6. Si vous retenez `criticalPolicy: asap`, attribuez à chaque sonde un décalage `security.criticalStagger` différent (§ 20.4.11).

Orchestration externe des activations (environnements critiques). Les sondes ne se coordonnent pas entre elles. Lorsque l'interruption simultanée de la supervision d'un site n'est pas acceptable, confiez l'enchaînement des activations à votre propre outil d'automatisation, avec l'API existante :

1. Placez le canal `security` des sondes du site en mode `manual` : aucune activation n'a lieu sans votre outil.
2. Activez la première sonde (`POST /api/v1/updates/apply`, § 20.4.3).
3. Conservez l'identifiant `activationId` renvoyé par la réponse de l'activation, ainsi que la version attendue (`staged.version` de `GET /api/v1/updates`, relevée avant l'activation).
4. Attendez son retour en service effectif, et non un simple délai. Interrogez `GET /api/v1/updates` à intervalles réguliers, jusqu'à un délai maximal que vous fixez d'après la durée totale observée d'une activation, retour arrière compris. La sonde n'est revenue en service que si **toutes** les conditions suivantes sont réunies :
   - `lastResult` n'est pas `null` et `lastResult.activationId` est égal à l'identifiant conservé à l'étape 3 ;
   - `lastResult.outcome` vaut `succeeded` ;
   - la version de l'emplacement actif (`slots` et `activeSlot`) est la version attendue ;
   - `state` ne vaut pas `maintenance` ;
   - `/health` répond `OK` (chapitre 6, § 6.6.2) ;
   - votre plateforme de supervision reçoit de cette sonde des résultats d'exécution postérieurs à `lastResult.finishedAt`.
5. Seulement alors, activez la sonde suivante.

Interprétez les autres situations ainsi :

Situation observée	Signification	Action
`lastResult` à `null`, ou `activationId` différent de celui de l'étape 3	Le résultat ne concerne pas votre activation : elle n'a pas encore produit de résultat, ou il s'agit d'une tentative antérieure	Continuer d'attendre, jusqu'au délai maximal
`outcome` à `in_progress`, ou `state` à `maintenance`	Activation en cours (drainage, redémarrage ou autotest)	Continuer d'attendre, jusqu'au délai maximal

Situation observée	Signification	Action
`outcome` à `rolled_back`	Retour arrière automatique : l'ancienne version est active	Arrêter l'enchaînement et traiter l'incident (chapitre 19, § 19.10)
`outcome` à `failed`	Échec avant redémarrage : l'emplacement actif est inchangé	Arrêter l'enchaînement et traiter l'incident
`outcome` à `succeeded`, mais version active différente de la version attendue	Incohérence : une autre image a été activée	Arrêter l'enchaînement et vérifier `GET /api/v1/updates` et le journal d'audit
Conditions réunies, sauf l'arrivée de résultats récents	La sonde a redémarré mais ne supervise pas, ou ses résultats ne parviennent pas à la plateforme	Ne pas poursuivre ; vérifier les destinations d'export et les contrôles de la sonde
Délai maximal dépassé	Sonde injoignable ou activation bloquée	Arrêter l'enchaînement ; intervenir sur la sonde (chapitre 19)

Un succès antérieur ne prouve jamais la reprise : `lastResult` désigne la **dernière tentative**, quelle que soit l'activation. La jauge `sva.update.last_result` peut servir de signal complémentaire dans vos tableaux de bord, mais elle ne porte ni la version ni l'identifiant d'activation : elle ne suffit pas à valider une étape d'orchestration. De même, le décalage `security.criticalStagger` (§ 20.4.11) réduit le risque d'activations simultanées, mais ne vérifie pas le retour des autres sondes.

> ! **Attention** — En mode `manual`, la sonde n'active jamais d'elle-même un correctif critique (§ 20.4.11). La priorité du correctif repose alors sur votre procédure : l'état de santé `updates` reste en `CRITICAL` et l'alerte « correctif critique en attente » (§ 20.7.1) reste active jusqu'à l'activation. Fixez dans votre procédure un délai maximal de traitement, afin que la coordination ne retarde jamais indéfiniment un correctif de sécurité.

> → **Évolution prévue** — La console de gestion centralisée prévue (chapitre 3, § 3.8) doit permettre d'orchestrer **séquentiellement** les mises à jour des sondes d'un même site : chaque activation ne serait engagée qu'après le retour vérifié de la précédente à un état sain, avec un délai maximal au-delà duquel un correctif critique serait activé malgré tout. Elle est prévue dans une version ultérieure ; la sonde ne dépendra jamais de cette console pour fonctionner.

Règles d'alerte de site. Deux alertes distinctes sont nécessaires, car un échec applicatif et une perte de couverture appellent des réactions différentes. Une sonde est dite **évaluable** pour un contrôle lorsqu'elle en fournit un résultat `OK`, `WARNING` ou `CRITICAL` et qu'elle n'est pas en maintenance.

Alerte	Se déclenche lorsque	Signification
`ParcoursEnEchecSurLeSite`	Toutes les sondes évaluables du site voient le contrôle en `CRITICAL`, au moins une sonde étant évaluable	Incident probable de l'application ou du parcours supervisé
`CouvertureInsuffisanteSurLeSite`	Le nombre de sondes évaluables est inférieur à l'effectif attendu du site, y compris lorsqu'aucune sonde ne fournit plus de résultat évaluable	Perte de couverture : la supervision du site repose sur moins de sondes que prévu, ou n'est plus assurée

Ces deux alertes forment le groupe de règles Prometheus `sondes-site`, fourni prêt à charger dans le fichier `exemples/prometheus/sva-regles-alerte.yml`. Leurs expressions PromQL, le seul paramètre à adapter (l'effectif attendu de chaque site, `sva:site_effectif_attendu`), leurs limites, l'extension facultative `sondes-site-controles` et l'adaptation au mode envoi (label `sva_site`) sont décrits au chapitre 18 (§ 18.15.6).

> **Information** — L'alerte de couverture se déclenche aussi pendant la maintenance planifiée d'une sonde, puisque celle-ci n'est plus évaluable. Choisissez pour elle un délai `for` supérieur à la durée totale mesurée d'une activation (§ 20.4.5), ou neutralisez cette seule alerte pendant les fenêtres des sondes du site (§ 20.4.10). Ne neutralisez jamais `ParcoursEnEchecSurLeSite` : c'est elle qui détecte un incident pendant la maintenance d'une sonde.

Résultat attendu : pendant la maintenance d'une sonde, la supervision du site reste assurée par les autres, et les alertes restent fondées sur leurs résultats.

Vérification : activez une mise à jour sur l'une des sondes (§ 20.4.3), ou attendez sa fenêtre, et vérifiez que l'autre continue de publier ses résultats et que la règle reste évaluée.

Erreurs possibles

Situation	Cause probable	Action
Alerte déclenchée pendant la maintenance d'une sonde	Règle fondée sur une seule sonde ; label `site` (ou `sva_site`) absent	Réécrire la règle sur l'ensemble des sondes ; vérifier les labels
Aucune supervision pendant une fenêtre	Fenêtres des sondes qui se chevauchent ; activation qui déborde sur la fenêtre de l'autre sonde ; activations manuelles simultanées	Décaler les fenêtres en tenant compte de la durée totale d'une activation (§ 20.4.8) ; activer les sondes l'une après l'autre
`CouvertureInsuffisanteSurLeSite` à chaque activation planifiée	Délai `for` inférieur à la durée d'une activation	Allonger le délai `for` ou neutraliser cette seule alerte pendant les fenêtres du site
Résultats divergents entre les sondes	Configurations différentes, ou accès réseau différents	Réimporter la configuration ; comparer les flux réseau des deux sondes (chapitre 7)

> **Information** — La durée pendant laquelle une sonde ne supervise pas lors d'une activation s'observe entre les événements `maintenance.started` et `maintenance.ended` (§ 20.4.5, § 20.4.9). Pour les sites qui ne disposent que d'une sonde, cette durée est la seule interruption planifiée de la supervision. Une activation **automatique** commence dans l'une des fenêtres que vous avez choisies, mais elle peut se prolonger au-delà de la fin de la fenêtre lorsque sa durée totale, retour arrière éventuel compris, dépasse le temps restant dans la fenêtre (§ 20.4.8). L'activation manuelle par un administrateur (§ 20.4.3) et l'activation d'un correctif critique avec `criticalPolicy: asap` (§ 20.4.11) peuvent en outre intervenir hors fenêtre. Sur un site redondant, l'espacement des fenêtres ne coordonne pas les activations `asap` : un même correctif critique publié pour toutes les sondes du site peut les activer à des instants proches, sauf si vous leur attribuez des décalages `security.criticalStagger` différents (§ 20.4.11), qui réduisent ce risque sans l'exclure.

20.5 Sauvegarde et restauration de la configuration

20.5.1 Sauvegarde automatique locale

La sonde sauvegarde automatiquement sa configuration une fois par jour et conserve les **trois dernières générations**. Cette sauvegarde lui sert à se restaurer elle-même si sa base de données est corrompue. Elle est stockée sur la même carte que la sonde : elle ne protège donc **pas** contre la perte ou la panne de la carte.

20.5.2 Exporter la configuration

Objectif : conserver hors de la sonde une copie de sa configuration, pour la restaurer sur une nouvelle carte ou une nouvelle sonde.

Prérequis : rôle `admin`.

Étapes

1. Lancez l'export depuis l'interface d'administration ou par l'API :

   ```bash
   sapi -o sonde-paris-01-config-2026-11-19.json \
        "$SONDE/api/v1/config/export"
   ```

   L'archive est un document JSON. Sans autre option, elle ne contient ni les comptes ni les secrets.

2. Choisissez le traitement des secrets :
   - **sans les secrets** (recommandé pour l'archivage courant) : il faudra les ressaisir lors d'une restauration ;
   - **avec les secrets chiffrés par une phrase de passe** : l'export se demande alors en `POST`, avec la phrase de passe dans l'en-tête `X-Config-Passphrase`, comme pour l'import, et jamais dans l'adresse, qui peut être journalisée. La phrase de passe est lue sans affichage et placée dans un fichier d'en-tête temporaire, lisible par vous seul et supprimé aussitôt (même méthode qu'au chapitre 9, § 9.12.2) :

     ```bash
     (
       umask 077
       ENTETE=$(mktemp) && trap 'rm -f "$ENTETE"' EXIT &&
       IFS= read -rs PHRASE &&
       printf 'X-Config-Passphrase: %s\n' "$PHRASE" > "$ENTETE" &&
       unset PHRASE &&
       sapi -X POST -H @"$ENTETE" \
         -o "sonde-paris-01-config-$(date +%Y%m%d).json" \
         "$SONDE/api/v1/config/export"
     )
     ```

     L'archive inclut les comptes (empreintes des mots de passe) et les secrets, chiffrés en AES-256-GCM avec une clé dérivée de la phrase de passe par Argon2id. À l'import, la phrase de passe est transmise de la même manière (§ 20.5.3). Conservez-la dans votre coffre-fort de mots de passe, jamais à côté de l'archive.

3. Rangez l'archive dans un espace de stockage sécurisé de votre entreprise, avec un nom indiquant la sonde et la date.

Résultat attendu : une archive contenant les contrôles, plans d'exécution, scripts, destinations d'export et paramètres, à l'exception de la section `network`. Les comptes et les secrets n'y figurent, chiffrés, que si une phrase de passe a été fournie.

> ✓ **Conseil** — Exportez la configuration après chaque modification importante et au moins une fois par mois. Pour les équipes qui gèrent la supervision « as code », l'archive ou ses composants peuvent être versionnés dans un dépôt de gestion de configuration, à condition d'en exclure les secrets.

20.5.3 Restaurer une configuration

1. Sur la sonde cible (mise en service effectuée, compte administrateur créé), validez d'abord l'archive **sans**

l'appliquer :

```bash
sapi -X POST \
     -H "Content-Type: application/json" \
     --data-binary @sonde-paris-01-config-2026-11-19.json \
     "$SONDE/api/v1/config/import?dryRun=true"
```

L'archive est transmise telle qu'elle a été exportée, au même format JSON. La réponse est un compte rendu JSON qui indique les éléments qui seraient créés ou modifiés, et les erreurs éventuelles (par exemple, un type de contrôle non couvert par l'édition de la sonde cible).

2. Corrigez les erreurs signalées, puis relancez l'import sans `dryRun=true` :

```bash
# Import effectif. Pour une archive chiffrée par une phrase de passe (§ 20.5.2),
# transmettez aussi l'en-tête X-Config-Passphrase par un fichier d'en-tête temporaire
# (-H @"$ENTETE"), créé comme pour l'export, jamais par une valeur en argument.
sapi -X POST \
     -H "Content-Type: application/json" \
     --data-binary @sonde-paris-01-config-2026-11-19.json \
     "$SONDE/api/v1/config/import"
```

3. Ressaisissez les secrets si l'archive n'en contenait pas (chapitre 9).
4. Vérifiez les destinations d'export avec leur fonction de test.
5. Lancez une exécution d'essai des contrôles principaux.

> **Attention** — L'archive ne contient pas la section `network` : les paramètres réseau de la sonde cible sont conservés. En revanche, l'identifiant de la sonde (`service.instance.id`) est restauré par défaut. Pour appliquer la configuration à une sonde destinée à fonctionner en parallèle de la sonde d'origine, ajoutez le paramètre `newIdentity=true` à la requête d'import, afin que les deux sondes restent distinctes dans les plateformes d'observabilité.

20.6 Remplacer une carte microSD ou une sonde

20.6.1 Cas d'usage

- Carte microSD défaillante ou en fin de vie (chapitre 19, § 19.3 et § 19.9).
- Raspberry Pi 5 défaillant.
- Remplacement préventif programmé.
- Changement d'édition : la nouvelle édition est livrée sur une nouvelle carte (§ 20.6.5). Si l'édition visée exige plus de mémoire, remplacez aussi le Raspberry Pi 5 (par exemple, un modèle de 8 Go pour l'édition Enterprise).

20.6.2 Procédure

Prérequis : une nouvelle carte microSD préparée par l'éditeur, de l'édition voulue, avec son encart (code d'installation). Chaque carte porte son propre certificat d'édition : une image écrite par vos soins sur une autre carte démarrerait en mode restreint (§ 6.5). Vous avez aussi besoin du dernier export de configuration (§ 20.5.2) et des secrets si l'export n'en contient pas. La carte se commande auprès de l'éditeur, selon les conditions de votre contrat.

Étapes

1. Si l'ancienne sonde fonctionne encore, exportez sa configuration et arrêtez-la proprement (§ 20.2.2).
2. Débranchez l'alimentation, retirez l'ancienne carte (chapitre 5) et étiquetez-la « à effacer ».

3. Insérez la nouvelle carte. Pour retrouver rapidement l'adresse et l'identité précédentes, déposez au préalable un fichier `provision.yaml` reprenant le nom d'hôte, le site et la configuration réseau d'origine (chapitre 6).
4. Mettez sous tension et effectuez la mise en service (chapitre 6).
5. Importez la configuration (§ 20.5.3) et ressaisissez les secrets si nécessaire.
6. Installez le certificat de l'entreprise si l'ancienne sonde en utilisait un (chapitre 7).
7. **Révoquez** les jetons d'API de l'ancienne sonde qui ne sont plus utilisés, puis créez-en de nouveaux pour Prometheus, Nagios ou Centreon et vos automatisations. Mettez à jour ces outils.
8. Vérifiez que les plateformes d'observabilité reçoivent de nouveau les données, et que la politique de mises à jour (section `updates`, restaurée par l'import) convient toujours : fenêtres de maintenance, et décalage par rapport aux autres sondes du site (§ 20.4.8, § 20.4.12).
9. Effacez de manière sécurisée l'ancienne carte (§ 20.13).

> **Information** — L'identifiant unique de la sonde (`service.instance.id` dans la télémétrie) est généré au premier démarrage. Lors de l'import de la configuration (étape 5), il est **restauré par défaut** : la sonde remplacée conserve son identifiant. La portée de cette continuité dépend du mode de transmission :
>
> - **en mode envoi (OTLP)**, `service.instance.id` restauré est une condition de la continuité des séries dans les plateformes d'observabilité, mais il ne suffit pas à l'assurer : l'identité finale d'une série dépend aussi des autres attributs de ressource exportés (nom d'hôte, site, version), de ceux que la destination promeut en labels ou en dimensions, et des règles propres à chaque destination. Vérifiez sur votre plateforme que les séries de la sonde remplacée se poursuivent ;
> - **en mode collecte (Prometheus)**, le label `instance` est dérivé de l'adresse de la cible collectée, et non de `service.instance.id` : les séries ne se poursuivent sous le même label que si la cible déclarée dans Prometheus reste identique : même nom d'hôte si la cible est déclarée par son nom, même adresse IP si elle l'est par son adresse (fichier de provisionnement, étape 3).
>
> Dans les deux cas, les **compteurs repartent de zéro** sur la nouvelle sonde ; les fonctions `increase()` et `rate()` de PromQL traitent cette réinitialisation sans erreur. **Mettez l'ancienne sonde hors service avant de réutiliser son identité** : deux sondes qui émettent le même identifiant en même temps mélangent leurs séries. Le nom d'hôte et le site (`sva.site`), repris par le fichier de provisionnement, complètent cette continuité. Si vous n'importez pas de configuration, ou si vous ajoutez le paramètre `newIdentity=true` (§ 20.5.3), la sonde apparaît sous un nouvel identifiant : prévoyez alors de supprimer l'ancienne instance de vos inventaires.

> **Information** — L'historique des exécutions n'est pas inclus dans l'export de configuration. Il est perdu lors d'un remplacement de carte. L'historique transmis aux plateformes d'observabilité reste disponible dans celles-ci. Si la sonde utilise un SSD (§ 20.3.4), l'historique est stocké sur celui-ci et non sur la microSD ; il n'est pas repris par la nouvelle carte : l'activation du SSD par celle-ci (§ 20.3.4) le reformate. Une réinitialisation usine efface également le SSD.

20.6.3 Carte de rechange

Pour les sondes critiques, conservez une carte de rechange de la même édition, acquise auprès de l'éditeur, accompagnée d'un `provision.yaml` prêt à l'emploi et du dernier export de configuration. Le remplacement prend alors quelques minutes. Le fichier de provisionnement contient des informations sensibles : conservez la carte de rechange dans un lieu sécurisé.

20.6.4 Remplacer une carte défectueuse

Le droit d'utilisation du logiciel intégré est attaché à la carte livrée. Si cette carte devient défectueuse, elle est **remplacée par une carte de la même édition, sans rachat du produit**. La nouvelle carte porte un nouveau certificat d'édition.

Prérequis : numéro de série du produit (encart, étiquette, ou état de santé `GET /api/v1/system` si la sonde démarre encore) ; description des symptômes (chapitre 19, § 19.3 et § 19.9) ; si possible, un export récent de la configuration (§ 20.5.2).

Étapes

1. Contactez le support (annexe L) en indiquant le numéro de série et les symptômes. Si la sonde démarre encore, joignez une archive de diagnostic (chapitre 19, § 19.12).
2. Après confirmation du défaut, le support vous adresse une carte de remplacement de la même édition, avec son encart.
3. Retournez la carte défectueuse selon les instructions du support, ou détruisez-la selon la procédure qu'il vous indique. Une carte défectueuse contient encore vos secrets chiffrés et leur clé de chiffrement. **Si elle ne peut plus être effacée** (carte illisible), vous n'êtes pas tenu de la retourner : vous pouvez la conserver dans un lieu sécurisé, ou la détruire physiquement et fournir au support une **attestation de destruction** à la place du retour. Le remplacement reste acquis dans ce cas. Pour toute carte retournée, joignez une preuve d'effacement (§ 20.13). Demandez au support le modèle d'attestation de destruction et ses modalités de transmission.
4. Installez la carte de remplacement en suivant la procédure du § 20.6.2.

Les conditions de remplacement, pendant et après la période de garantie, sont celles de votre contrat.

> ✓ **Conseil** — La rapidité d'un remplacement dépend surtout de la fraîcheur de votre dernier export de configuration. Exportez la configuration après chaque modification importante (§ 20.5).

20.6.5 Changer d'édition

Une édition supérieure est livrée sur une **nouvelle carte microSD**. Votre configuration, vos scénarios et vos secrets y sont transférés par un export suivi d'un import. L'ancienne carte reste un produit utilisable dans son édition d'origine, sauf si vous la restituez dans le cadre d'une offre de reprise, selon les conditions de votre contrat.

Prérequis : la carte de la nouvelle édition et son encart ; rôle `admin` sur la sonde actuelle ; un Raspberry Pi 5 dont la capacité mémoire convient à la nouvelle édition (chapitre 4, § 4.1.2).

> ℹ **Information** — Si la sonde utilise un SSD NVMe (§ 20.3.4), il reste en place : le changement d'édition ne concerne que la carte microSD, qui porte le certificat d'édition, la configuration et les secrets. L'historique des exécutions, stocké sur le SSD, suit la règle du § 20.6.2.

Étapes

1. Exportez la configuration **avec une phrase de passe**, afin d'inclure les comptes et les secrets, chiffrés (§ 20.5.2). Choisissez une phrase de passe longue et conservez-la dans votre coffre-fort de mots de passe, séparément de l'archive.
2. Arrêtez proprement la sonde (§ 20.2.2), puis remplacez la carte. Pour retrouver le même nom d'hôte et la même adresse, déposez au préalable sur la nouvelle carte un `provision.yaml` reprenant ces paramètres (chapitre 6).
3. Mettez la sonde sous tension et terminez la mise en service. Vérifiez que l'édition affichée est la nouvelle édition (chapitre 6, § 6.5).
4. Validez l'import sans l'appliquer (`dryRun=true`), en transmettant la phrase de passe dans l'en-tête `X-Config-Passphrase` (§ 20.5.3), puis appliquez-le. L'identifiant de la sonde est restauré ; cette restauration ne suffit pas, à

elle seule, à assurer la continuité des séries de métriques dans vos plateformes : ses conditions sont précisées au § 20.6.2 (encadré qui suit la procédure). Vérifiez sur vos plateformes que les séries de la sonde se poursuivent.

5. Créez de nouveaux jetons d'API pour Prometheus, Nagios, Centreon et vos automatisations : les jetons ne font pas partie de l'export. Mettez à jour ces outils.
6. Vérifiez les destinations d'export avec leur fonction de test, puis lancez une exécution d'essai des contrôles principaux.
7. Supprimez l'archive d'export des emplacements temporaires utilisés pour le transfert.
8. Si vous restituez l'ancienne carte, réinitialisez-la d'abord aux paramètres d'usine (§ 20.10) afin qu'elle ne contienne plus aucune donnée de votre organisation, puis suivez les instructions de restitution.

Résultat attendu : la sonde fonctionne avec la nouvelle édition, avec la même configuration, le même identifiant et les mêmes secrets. Les fonctions propres à la nouvelle édition sont disponibles.

> **Avertissement** — L'archive exportée avec une phrase de passe contient vos secrets, chiffrés. Ne la transmettez jamais par courriel ou par messagerie, et ne conservez jamais la phrase de passe à côté de l'archive.

20.7 Surveillance de la santé de la sonde

20.7.1 Indicateurs à suivre

La sonde publie des métriques sur sa propre santé (chapitre 18). Les alertes suivantes sont recommandées dans votre plateforme de supervision.

Alerte	Condition indicative	Signification
Sonde muette	Absence de `sva.agent.up` pendant 5 minutes	Sonde éteinte, injoignable ou export interrompu (équivalent de l'état `STALE` de la sonde)
Sous-tension	`sva.hw.throttled` égal à 1 avec l'attribut `sva.hw.reason` = `undervoltage`	Alimentation insuffisante : risque de corruption
Surchauffe	`sva.hw.throttled` égal à 1 avec `sva.hw.reason` = `thermal`, ou `sva.hw.temperature` durablement élevée	Refroidissement à vérifier
Stockage	Occupation d'un système de fichiers de la sonde (`system.filesystem.usage`, état `used`) au-delà de 80 % de sa capacité pendant 30 minutes ; le détail par zone est donné par `sva.storage.usage`	Rétention à réduire ou support à remplacer (§ 20.6.4)
Saturation	`sva.agent.scheduler.lag` en hausse durable	Trop de contrôles pour la sonde
Échecs de Chromium	Hausse de `sva.browser.launches` en échec	Problème du moteur de scénarios
Export	`sva.export.failures` ou `sva.export.dropped` en hausse	Destination d'export en erreur
Récupération de base	Incrément de `sva.storage.recoveries`	Coupure ayant corrompu la base : examiner l'alimentation
Mise à jour en attente sans fenêtre	État de santé `updates` en `WARNING`, ou `sva.update.pending` égal à 1 depuis plus de 24 heures sans maintenance planifiée (`sva.maintenance.next_start` égal à `0` ; une série absente traduit une perte de collecte, pas l'absence de planification)	Aucune fenêtre de maintenance : la mise à jour ne sera pas activée sans intervention (§ 20.4.2, § 20.4.8)

Alerte	Condition indicative	Signification
Correctif critique en attente	État de santé `updates` en `CRITICAL`, ou `sva.update.pending` égal à 1 avec l'attribut `sva.update.critical` = `true`	Vulnérabilité activement exploitée : décider de l'activation (§ 20.4.11)
Mise à jour en échec	`sva.update.last_result` porte `rolled_back` ou `failed` (`lastResult` de `GET /api/v1/updates`)	Retour arrière automatique ou échec avant redémarrage : la version précédente reste active ; consulter le journal d'audit et le chapitre 19, § 19.10
Maintenance anormalement longue	`sva.agent.maintenance` égal à 1 au-delà de la durée de la fenêtre, ou sonde muette après la fin de la fenêtre de maintenance (`sva.maintenance.next_end` dépassé) ou, pour une activation sans fin prévue (`sva.maintenance.next_end` à `0`), après la durée maximale que vous avez retenue	Activation bloquée ou sonde non revenue après son redémarrage : consulter le journal d'audit et le chapitre 19, § 19.10

Les noms des métriques utilisent le préfixe stable `sva.`. Leur traduction au format Prometheus et des exemples de requêtes figurent au chapitre 18.

20.7.2 Vérification manuelle

En complément des alertes, consultez périodiquement la page de santé de l'interface ou `GET /api/v1/system` : version, ressources, état des modules et des destinations d'export. Trois états de santé concernent particulièrement la maintenance : `storage`, qui passe en `CRITICAL` si le SSD activé est absent ou défaillant (§ 20.3.4) ; `support`, qui passe en `WARNING` à la fin de la période de support (§ 20.4.6) ; `updates`, qui passe en `WARNING` lorsqu'une mise à jour attend sans activation planifiée ou après un report (§ 20.4.2) et en `CRITICAL` tant qu'un correctif critique n'est pas actif (§ 20.4.11).

20.8 Rotation des secrets et des jetons

Élément	Rotation recommandée	Procédure
Mots de passe des comptes de test des applications	Selon la politique de l'entreprise, et au moins une fois par an	Changer le mot de passe dans l'application, puis remplacer immédiatement le secret dans la sonde (`PUT /api/v1/secrets/{name}`). Le secret ne peut jamais être relu, seulement remplacé
Comptes des bases de données (lecture seule)	Idem	Idem
Clés d'ingestion des plateformes (Datadog, Dynatrace, Grafana)	Selon la politique de la plateforme	Créer la nouvelle clé, mettre à jour le secret, tester la destination, puis révoquer l'ancienne clé
Jetons d'API de la sonde	Au moins une fois par an ; immédiatement au départ d'un collaborateur ou en cas de doute	Créer un nouveau jeton, mettre à jour l'outil qui l'utilise, révoquer l'ancien
Mots de passe des comptes de l'interface	Selon la politique de l'entreprise	Changement par l'utilisateur ou par un administrateur
Certificat TLS de la sonde	Avant expiration (certificat émis par l'autorité de l'entreprise)	Téléverser le nouveau certificat (chapitre 7)

> ✓ **Conseil** — Pour éviter qu'une rotation de mot de passe ne provoque une alerte, coordonnez-la avec la mise à jour du secret : changez le mot de passe dans l'application, puis mettez à jour le secret dans les minutes qui suivent. La date de rotation de chaque secret est visible dans l'interface. Consultez-la lors de la revue trimestrielle (§ 20.11).

> ! **Avertissement** — Lorsqu'un secret est supprimé ou remplacé, révoquez également l'ancienne valeur dans le système qui l'a émise. Supprimer un secret de la sonde ne le rend pas invalide pour l'application concernée.

20.9 Maintenance des scénarios et des contrôles

Les applications supervisées évoluent : nouvelles versions, changement de charte graphique, nouveaux parcours, nouvelles règles de sécurité. Un scénario non entretenu finit par échouer alors que l'application fonctionne. Il produit alors de fausses alertes et décrédibilise la supervision.

20.9.1 Intégrer la supervision au cycle de livraison

1. **Avant une mise en production**, l'équipe applicative informe le responsable de la supervision des changements visibles : libellés, structure des pages, parcours, authentification.
2. **Sur l'environnement de recette**, une sonde (ou un contrôle dédié) exécute les scénarios sur la nouvelle version. Les écarts sont corrigés avant la mise en production.
3. **Pendant la mise en production**, les contrôles concernés sont suspendus ou placés en période de maintenance, si la plateforme le permet.
4. **Après la mise en production**, une exécution d'essai de chaque scénario concerné confirme son fonctionnement.

> ✓ **Conseil** — Demandez aux équipes de développement d'ajouter des attributs stables dédiés aux tests (`data-testid`) sur les éléments utilisés par les scénarios. C'est la mesure qui réduit le plus la maintenance des scénarios (chapitre 16).

20.9.2 Revue périodique des contrôles

Une fois par trimestre, passez en revue l'ensemble des contrôles :

- contrôles désactivés depuis longtemps : à supprimer ou à réactiver ;
- contrôles en bagotage fréquent : seuils ou nouvelles tentatives à ajuster (chapitre 19, § 19.6) ;
- scénarios dont la durée augmente régulièrement : dérive de performance à signaler ;
- applications retirées du service : contrôles à supprimer ;
- nouvelles applications : contrôles à créer ;
- comptes de test : toujours valides, droits minimaux, pas de données réelles.

20.10 Réinitialisation aux paramètres d'usine

La réinitialisation efface la configuration, les résultats, les comptes, les jetons et les secrets, puis ramène la sonde à l'état « non initialisée » : le code d'installation imprimé sur l'encart ou l'étiquette du produit redevient valide pour une nouvelle mise en service. Le certificat d'édition est conservé.

Méthode	Description
Par l'interface ou l'API	Action réservée au rôle `admin` : `POST /api/v1/system/factory-reset` avec le corps `{"confirm": "<nom d'hôte de la sonde>"}`. La confirmation par le nom d'hôte évite une réinitialisation de la mauvaise sonde.
Sans accès administrateur	Procédure physique par fichier déposé sur la partition `SVA-BOOT` de la microSD, confirmée en deux temps par un code aléatoire (§ 20.10.1). Si seul le mot de passe administrateur est perdu, préférez la récupération sans perte de configuration (§ 20.10.2)
Réécriture de la carte	Remplacement par une carte neuve préconfigurée (§ 20.6)

> **Avertissement** — La réinitialisation est irréversible. Exportez la configuration au préalable si elle doit être réutilisée. Pensez à révoquer, côté applications et plateformes, les identifiants qui étaient stockés sur la sonde si celle-ci change de main.

> **Information** — La réinitialisation usine lève le verrouillage définitif de SSH (`ssh.locked` , chapitre 7) : c'est le seul moyen de le lever. Après la réinitialisation, SSH est de nouveau désactivé par défaut, comme à la sortie de l'emballage.

20.10.1 Réinitialiser sans accès administrateur (procédure physique)

Objectif : ramener aux paramètres d'usine une sonde dont aucun compte administrateur n'est plus utilisable, par exemple une sonde récupérée sans ses identifiants.

La procédure exige un **accès physique** à la carte microSD et une **confirmation en deux temps** : le premier démarrage ne fait que produire un code aléatoire, et rien n'est effacé tant que ce code n'a pas été recopié. Le numéro de série de la sonde, qui figure sur l'étiquette et l'encart, ne suffit jamais à autoriser une réinitialisation.

Prérequis : accès physique à la sonde ; un poste de travail capable de lire la partition de démarrage de la microSD, au format FAT32 et étiquetée `SVA-BOOT` (chapitre 6, § 6.3.5) ; facultativement, un écran raccordé en micro-HDMI.

> **Avertissement** — Avant d'engager cette procédure, vérifiez si la récupération du compte administrateur (§ 20.10.2) est possible : elle conserve toute la configuration. La réinitialisation, elle, efface définitivement la configuration, les résultats, les comptes, les jetons et les secrets.

Étapes — premier temps : demande

1. Arrêtez la sonde par un appui bref sur le bouton d'alimentation (§ 20.2.2), attendez l'extinction, puis débranchez l'alimentation.
2. Retirez la carte microSD (chapitre 5) et ouvrez sur votre poste sa partition de démarrage, étiquetée `SVA-BOOT`. Avec le mécanisme de mise à jour A/B, d'autres partitions de la carte peuvent apparaître sur le poste : ne les modifiez pas. Ne modifiez aucun autre fichier de la partition `SVA-BOOT`.
3. Créez à la racine de cette partition un fichier texte nommé `factory-reset.txt` (le fichier peut être vide : son contenu est ignoré à ce stade). Vérifiez que votre système n'a pas ajouté d'extension masquée (par exemple `factory-reset.txt.txt`).

4. Éjectez proprement la carte, réinsérez-la dans la sonde et mettez celle-ci sous tension.
5. Au démarrage, la sonde **n'efface rien**. Elle écrit sur la partition `SVA-BOOT` un fichier `factory-reset-challenge.txt` contenant un **code aléatoire de 8 caractères**, valable **24 heures**, et un avertissement sur les conséquences de l'opération. Elle affiche aussi ce code sur la console HDMI et consigne la demande dans le journal.

Étapes — second temps : confirmation

6. Relevez le code sur la console HDMI ou, après un nouvel arrêt propre et le retrait de la carte, dans le fichier `factory-reset-challenge.txt`. Lisez l'avertissement qu'il contient.
7. Remplacez le contenu de `factory-reset.txt` par la seule ligne suivante, où `<code>` est le code relevé :

   ```
   EFFACER <code>
   ```
   TEXTE

8. Réinsérez la carte et redémarrez la sonde avant l'expiration du délai de 24 heures.

Résultat attendu : si le code est exact et encore valide, la sonde se réinitialise et revient à l'état « non initialisée ». Le certificat d'édition est conservé et le code d'installation imprimé sur l'encart ou l'étiquette redevient valide. Le verrouillage de SSH éventuel est levé.

Vérification : la sonde attend une nouvelle mise en service ; la console HDMI et le fichier `setup-code.txt` de la partition `SVA-BOOT` affichent de nouveau le code d'installation (chapitre 6, § 6.4.3).

Erreurs possibles

Situation	Conséquence	Action
Code inexact	Aucune réinitialisation	Relire le code dans `factory-reset-challenge.txt` ou sur la console HDMI ; corriger la ligne `EFFACER <code>`
Code expiré (plus de 24 heures)	Aucune réinitialisation	Recommencer la procédure au premier temps pour obtenir un nouveau code
Fichier mal nommé ou placé dans un sous-dossier	La demande n'est pas prise en compte	Placer `factory-reset.txt` à la racine de la partition `SVA-BOOT`, sans extension supplémentaire ; ne pas l'écrire sur une autre partition de la carte

Précautions : la carte microSD contient vos secrets chiffrés et leur clé (§ 20.13). Ne la confiez pas à un tiers pour réaliser cette procédure. Après la réinitialisation, procédez à une nouvelle mise en service (chapitre 6) ou, si la sonde quitte votre organisation, poursuivez avec le § 20.13.

20.10.2 Récupérer l'accès administrateur sans perte de configuration

Objectif : définir un nouveau mot de passe pour un compte `admin` existant lorsque plus aucun administrateur ne peut se connecter (départ d'un collaborateur, mot de passe perdu), **sans perdre la configuration** de la sonde.

Prérequis

- Accès physique à la sonde et un poste capable de lire la partition de démarrage de la microSD, au format FAT32 et étiquetée `SVA-BOOT`.
- Le **code d'installation** imprimé sur l'encart ou l'étiquette du produit.
- Le **code de récupération** : code de 24 caractères, à usage unique, généré à la fin de la mise en service et affiché une seule fois, dans l'assistant ou, pour une sonde provisionnée par fichier, à la première connexion (chapitre 6, § 6.4.6). Il doit avoir été conservé dans le coffre-fort de mots de passe de votre organisation. La sonde n'en conserve qu'une empreinte (Argon2id) : ni la sonde ni le support de l'éditeur ne peuvent le retrouver.
- L'adresse de la sonde sur le réseau (chapitre 6, § 6.4.2).

> **Attention** — Sans code de récupération, cette procédure est impossible : seule la réinitialisation usine reste disponible (§ 20.10.1), avec perte de la configuration. Conservez le code de récupération et l'encart dès la mise en service.

Étapes

1. Arrêtez la sonde par un appui bref sur le bouton d'alimentation (§ 20.2.2), puis débranchez l'alimentation.
2. Retirez la carte microSD et créez, à la racine de sa partition de démarrage étiquetée `SVA-BOOT`, un fichier texte nommé `recovery.txt` (le fichier peut être vide : son contenu est ignoré). D'autres partitions de la carte peuvent apparaître sur le poste : ne les modifiez pas.
3. Réinsérez la carte et mettez la sonde sous tension.
4. Au démarrage, la sonde ouvre une **page de récupération** pendant **30 minutes** (`https://<adresse de la sonde>/recovery`). Connectez-vous-y depuis un poste du réseau où la sonde est installée, en vérifiant le certificat de la sonde comme au chapitre 6 (§ 6.4.4).
5. Saisissez le **code d'installation** et le **code de récupération**. Vous disposez de **3 essais au plus** par ouverture de la page.
6. Choisissez le compte `admin` existant à récupérer et définissez son nouveau mot de passe.
7. **Relevez le nouveau code de récupération** affiché par la page : l'ancien vient d'être consommé. Rangez-le immédiatement dans votre coffre-fort de mots de passe. Il ne sera plus affiché.
8. Connectez-vous à l'interface avec le compte récupéré.

Résultat attendu

- Le compte `admin` dispose de son nouveau mot de passe ; toutes ses sessions et tous ses jetons d'API ont été révoqués.
- La configuration, les contrôles, les secrets, les résultats et les autres comptes sont **conservés**.
- Un nouveau code de récupération a été émis ; le fichier `recovery.txt` a été supprimé par la sonde.

Vérification : le journal d'audit contient l'événement de récupération ; `recovery.txt` n'est plus présent sur la partition `SVA-BOOT` ; les contrôles s'exécutent normalement.

Erreurs possibles

Situation	Cause probable	Action
Page de récupération inaccessible	Délai de 30 minutes écoulé ; fichier mal nommé ou mal placé	Redéposer `recovery.txt` à la racine de la partition `SVA-BOOT` (et non sur une autre partition de la carte) et redémarrer la sonde
Codes refusés, page fermée après 3 essais	Faute de frappe ; code d'une autre sonde ; code de récupération déjà consommé	Vérifier l'encart de **cette** sonde et le dernier code de récupération conservé ; recommencer depuis l'étape 1
Code de récupération introuvable	Code non conservé lors de la mise en service	Réinitialisation usine (§ 20.10.1)

Précautions

- Recréez les jetons d'API utilisés par vos outils (Prometheus, Nagios, Centreon, automatisations) s'ils appartenaient au compte récupéré, puis mettez ces outils à jour.
- Si la perte d'accès fait suite au départ d'un collaborateur, revoyez aussi les autres comptes et jetons (chapitre 9).
- La récupération reste possible même lorsque SSH a été verrouillé définitivement (chapitre 7).

20.11 Plan de maintenance préventive

Fréquence	Opération	Référence
Quotidienne (automatisée)	Vérifier par des alertes que la sonde émet (`sva.agent.up`) et que les exports fonctionnent	§ 20.7
	Traiter les contrôles en `CRITICAL`, `UNKNOWN` ou `STALE`	Chapitres 17 et 19
Hebdomadaire	Consulter la page de santé : ressources, température, sous-tension, stockage	§ 20.7.2
	Vérifier les mises à jour disponibles ou en attente (canal en mode manuel, absence de fenêtre, report en cours), les activations annoncées, et lire les notes de version	§ 20.4
Mensuelle	Exporter la configuration hors de la sonde	§ 20.5.2
	Examiner les tendances de durée des scénarios principaux	Chapitre 17
	Consulter le journal d'audit à la recherche d'opérations inattendues	Chapitre 9
Trimestrielle	Revue des contrôles et des scénarios	§ 20.9.2
	Revue des comptes de l'interface, des rôles et des jetons d'API (révoquer l'inutile)	Chapitre 9
	Vérifier l'ancienneté des secrets	§ 20.8
	Vérifier la date d'expiration du certificat de la sonde	Chapitre 7
	Revoir les fenêtres de maintenance, leur alignement avec les périodes de maintenance des plateformes et leur décalage entre sondes d'un même site	§ 20.4.8, § 20.4.10, § 20.4.12
Annuelle	Rotation des secrets et des jetons	§ 20.8
	Test de restauration de la configuration sur une carte de rechange	§ 20.5.3, § 20.6
	Inspection physique et dépoussiérage	§ 20.12
	Évaluation du remplacement préventif de la carte microSD selon l'usage	§ 20.3.3
	Vérifier les dates de fin de la période de support (`supportEndsAt`) et du justificatif de maintenance (`maintenanceEndsAt`), et la conservation du code de récupération	§ 20.4.6, § 20.4.7, § 20.10.2

20.12 Entretien physique

> **Avertissement** — Manipulez la carte Raspberry Pi 5 par ses bords, après vous être déchargé de l'électricité statique (contact avec une partie métallique reliée à la terre). Ne touchez pas les composants ni les contacts de la carte microSD.

1. Arrêtez la sonde proprement avant toute intervention physique (§ 20.2.2).
2. Vérifiez que les aérations du boîtier ne sont pas obstruées et que le ventilateur tourne librement.
3. Retirez la poussière à l'air sec (bombe d'air pour matériel électronique), en immobilisant le ventilateur pour ne pas le faire tourner à grande vitesse. N'utilisez ni liquide ni aspirateur.
4. Vérifiez l'état du câble d'alimentation, du connecteur USB-C et du câble Ethernet : absence de pliure, de traction ou d'échauffement.
5. Vérifiez que la sonde reste fixée et protégée des chocs, de l'humidité et des sources de chaleur.

20.13 Fin de vie et effacement sécurisé

La carte microSD et, le cas échéant, le SSD d'une sonde contiennent des informations sensibles : secrets chiffrés et leur clé de chiffrement, configuration du réseau interne, noms et adresses des applications supervisées, captures d'écran, historique. Une sonde retirée du service doit être traitée comme un support de données confidentielles.

Étapes

1. Exportez la configuration si elle doit être réutilisée (§ 20.5.2).
2. Révoquez, côté applications et plateformes, tous les identifiants stockés sur la sonde : comptes de test, comptes de base de données, clés d'ingestion. Révoquez aussi les jetons d'API de la sonde utilisés par vos outils.
3. Effectuez une réinitialisation usine (§ 20.10). Elle ne garantit pas, à elle seule, l'effacement physique des données sur une mémoire flash.
4. Effacez ensuite le support selon la politique de sécurité de votre entreprise. Sur une mémoire flash (microSD, SSD), un simple formatage ne suffit pas. En cas de doute, la **destruction physique** de la carte est la méthode la plus sûre. Pour un SSD NVMe, utilisez la fonction d'effacement sécurisé du constructeur.
5. Retirez la sonde de vos inventaires, des cibles Prometheus, des services Nagios ou Centreon et des tableaux de bord.
6. Recyclez le matériel électronique via une filière agréée (déchets d'équipements électriques et électroniques).

> **Information** — Pour un support retourné à l'éditeur (remplacement, reprise, fin de prêt), joignez une **preuve d'effacement** conforme à la politique de votre entreprise, ou, si le support a été détruit, une **attestation de destruction**. Une carte défectueuse impossible à effacer n'a pas à être retournée : vous pouvez la conserver ou la détruire physiquement, et son remplacement reste acquis (§ 20.6.4).

> **Avertissement** — Ne revendez pas et ne réaffectez pas une carte microSD de sonde sans effacement sécurisé préalable. Les secrets y sont chiffrés, mais la clé de chiffrement se trouve sur le même support (chapitres 7 et 9).

ANNEXES

Références, glossaire et informations pratiques

Chapitre 21 — Annexes

Chapitre 21 — Annexes

Ces annexes regroupent les informations de référence du manuel : vocabulaire, questions fréquentes, listes de vérification, tableaux de référence, assistance et mentions légales. Elles sont conçues pour être consultées directement, sans relire les chapitres concernés, auxquels elles renvoient pour le détail.

Annexe	Contenu
A	Glossaire
B	Questions fréquentes
C	Récapitulatif des prérequis
D	Checklist de mise en service
E	Tableau des flux réseau
F	Référence de l'API REST
G	Référence des configurations
H	Types d'erreur
I	États et codes retour Nagios
J	Correspondance avec la génération historique
K	Informations de version et statut des informations
L	Assistance et contacts
M	Licences tierces et mentions

21.1 Annexe A — Glossaire

Les termes sont classés par ordre alphabétique. Les termes en police `code` désignent des valeurs ou des noms techniques tels qu'ils apparaissent dans l'API et la configuration.

Terme	Définition
802.1X	Norme de contrôle d'accès au réseau : l'équipement doit s'authentifier auprès du commutateur pour obtenir l'accès au port. La sonde la prend en charge sur sa prise Ethernet avec la méthode EAP-TLS (authentification par certificat), configurée dans la section `dot1x`. L'exception par adresse MAC (MAB) reste possible en alternative. Voir chapitre 7
A/B (emplacements système)	Organisation de la microSD en deux emplacements système, l'un actif, l'autre inactif, en lecture seule. Une mise à jour s'installe sur l'emplacement inactif ; la sonde redémarre une fois sur ce nouvel emplacement et revient automatiquement à l'emplacement précédent si elle n'est pas saine dans les 10 minutes. Voir chapitre 20
Agent Datadog	Logiciel de collecte de Datadog installé sur l'infrastructure du client. Il peut recevoir des données OpenTelemetry (récepteur OTLP) et les transmettre à Datadog. Voir chapitre 18
Agrégat horaire	Synthèse calculée par la sonde pour chaque contrôle et chaque heure (nombre d'exécutions, d'échecs, durées médiane et au 95e centile). Les agrégats sont conservés plus longtemps que les exécutions détaillées

Terme	Définition
API REST	Interface de programmation accessible en HTTPS, qui permet de piloter la sonde par des requêtes au format JSON. Base : `/api/v1`. Voir chapitre 9 et annexe F
APM (*Application Performance Monitoring*)	Supervision des performances applicatives à partir du trafic réel, au moyen d'agents installés sur les serveurs. Complémentaire de la supervision synthétique. Voir chapitre 2
Artefact	Fichier produit par une exécution, principalement en cas d'échec : capture d'écran, journal de console du navigateur, extrait réseau. Conservé sur la sonde avec un quota
Assertion	Condition vérifiée par un contrôle ou un scénario (code HTTP attendu, texte présent, valeur d'un champ JSON). Une assertion non vérifiée fait échouer l'exécution avec le type d'erreur `assertion_failed`
Assistant de mise en service	Suite d'écrans de l'interface d'administration qui guide la première configuration de la sonde. Il est protégé par le code d'installation. Voir chapitre 6
Attribut	Information associée à une métrique, une trace ou un log (par exemple, l'identifiant du contrôle). Le nombre de valeurs distinctes d'un attribut détermine la cardinalité
Authentification multifacteur (MFA)	Authentification qui combine plusieurs facteurs, par exemple un mot de passe et un code à usage unique. La sonde ne la contourne pas : un compte de test adapté est nécessaire. Voir chapitres 16 et 19
Autorité de certification (AC)	Organisme ou service qui signe des certificats TLS. Une entreprise dispose souvent de sa propre autorité interne, que la sonde doit connaître pour vérifier les certificats des applications internes
Bac à sable (sandbox)	Mécanisme d'isolation qui limite ce qu'un programme peut faire sur le système. Chromium utilise son propre bac à sable, conservé par la sonde
Bagotage (*flapping*)	Alternance rapide d'un contrôle entre réussite et échec. Il traduit souvent un problème intermittent ou un seuil mal réglé. Voir chapitres 8 et 17
Bridage (*throttling*)	Réduction automatique de la fréquence du processeur du Raspberry Pi en cas de surchauffe ou de sous-tension. Signalé par `vcgencmd get_throttled` et la métrique `sva.hw.throttled`
CAPTCHA	Test destiné à distinguer un humain d'un programme. Un scénario ne doit jamais chercher à le contourner : demandez une exemption pour la sonde. Voir chapitre 16
Cardinalité	Nombre de combinaisons distinctes de valeurs d'attributs pour une métrique. Une cardinalité élevée augmente le coût et la charge des plateformes d'observabilité
Carte de démonstration	Carte microSD prêtée par l'éditeur pour une évaluation, distincte du produit vendu. Son certificat d'édition est limité à une période d'évaluation de 30 jours, au terme de laquelle la sonde passe en mode restreint ; la carte est ensuite restituée. Cette expiration ne concerne jamais une carte vendue. Voir chapitre 6, § 6.5
Centile (percentile)	Valeur en dessous de laquelle se situe un pourcentage donné des mesures. Le 95e centile (p95) d'une durée est la durée que 95 % des exécutions ne dépassent pas
Centreon	Solution de supervision compatible avec les plugins Nagios. Elle peut interroger la sonde avec le plugin `check_sva`
Certificat autosigné	Certificat TLS signé par la sonde elle-même, généré au premier démarrage. Le navigateur affiche un avertissement tant qu'il n'est pas reconnu. On vérifie alors son empreinte
Certificat d'édition	Fichier signé par l'éditeur, inscrit sur la carte microSD lors de sa préparation et lié à l'identifiant matériel de cette carte. Il détermine l'édition de la sonde, est vérifié localement et n'expire pas. Voir chapitre 6, § 6.5
`check_sva`	Plugin Nagios qui interroge l'état d'un contrôle de la sonde depuis Nagios ou Centreon, sous forme de contrôle actif. Voir chapitre 18
Chrome DevTools Protocol (CDP)	Interface de pilotage de Chromium, utilisée par Puppeteer pour commander le navigateur. Voir chapitre 13

Terme	Définition
Chromium	Navigateur web libre, base de Google Chrome. La sonde l'utilise sans interface graphique (mode headless) pour exécuter les scénarios. Il est présent sur la carte dans toutes les éditions, puisqu'elles partagent le même système, mais il est désactivé par le certificat d'édition hors édition Enterprise
CID (*Card Identification*)	Identifiant matériel inscrit par le fabricant dans chaque carte microSD. Le certificat d'édition est lié au CID de la carte fournie : une copie du système sur une autre carte démarre en mode restreint. Voir **Certificat d'édition**
Classe d'exécution	Catégorie de contrôles partageant une limite de concurrence : contrôles légers, plugins et bases de données, navigateur. Voir chapitre 8
Code d'installation	Code propre à chaque carte, généré lors de sa préparation et imprimé sur l'encart ou l'étiquette. Il protège l'assistant, est invalidé à la fin de celui-ci et redevient valide après une réinitialisation usine ; il est aussi exigé pour la récupération du compte administrateur. Voir chapitres 6 et 20
Code de récupération	Code de 24 caractères, généré à la fin de la mise en service et affiché une seule fois, à conserver dans le coffre-fort de l'organisation. Associé au code d'installation et à un accès physique à la carte, il permet de redéfinir le mot de passe d'un compte `admin` sans perte de configuration. Il est consommé à chaque utilisation et remplacé par un nouveau code. Voir chapitres 6 et 20
Collecte (*scrape*)	Interrogation périodique de l'adresse `/metrics` de la sonde par Prometheus (mode *pull*). Voir chapitre 18
Collecteur OpenTelemetry (*OpenTelemetry Collector*)	Logiciel libre qui reçoit, transforme et réachemine des données OpenTelemetry. Il est souvent déployé comme passerelle par les clients. Voir chapitre 18
Compte de test	Compte applicatif dédié à la supervision, sans données réelles ni droits superflus, utilisé par les scénarios et contrôles
Concurrence	Nombre d'exécutions menées simultanément. Elle est limitée par classe d'exécution pour préserver les ressources de la sonde
Console centrale	Outil facultatif de gestion de plusieurs sondes, hébergé par l'éditeur ou chez le client. Évolution prévue
Contrôle	Unité de supervision configurée sur la sonde : un type (`http`, `browser`…), une cible, une planification et des critères de réussite
Contrôle actif / passif	Dans l'écosystème Nagios, un contrôle actif est lancé par le serveur de supervision ; un contrôle passif lui est transmis par un tiers. Le plugin `check_sva` est un contrôle actif
Cron (expression)	Syntaxe de planification en cinq champs (minute, heure, jour du mois, mois, jour de la semaine), par exemple `*/15 8-20 * * *`
CSR (*Certificate Signing Request*)	Demande de signature de certificat. La sonde la génère avec une clé privée qui ne la quitte jamais (`ec-p256` par défaut, ou `rsa-2048`), puis l'autorité de certification de l'entreprise la signe. Utilisée pour l'identifiant 802.1X et pour le certificat de l'interface d'administration. Voir chapitre 7
Cumulative (temporalité)	Mode de transmission d'une métrique où chaque valeur représente le total depuis le démarrage. Attendu par Prometheus
`data-testid`	Attribut HTML ajouté par les équipes de développement pour désigner de façon stable un élément utilisé par les tests et les scénarios. C'est le sélecteur à privilégier. Voir chapitre 16
Datadog	Plateforme d'observabilité en mode SaaS. Elle reçoit les données de la sonde via OTLP. Voir chapitre 18
Delta (temporalité)	Mode de transmission d'une métrique où chaque valeur représente la variation depuis l'envoi précédent. Exigé par Dynatrace et par l'ingestion OTLP directe de Datadog
Destination d'export	Plateforme qui reçoit la télémétrie de la sonde : backend OTLP, Datadog, Dynatrace, Grafana, Prometheus. Objet `exporter` de l'API
DHCP	Protocole d'attribution automatique d'adresse IP. Mode réseau par défaut de la sonde

Terme	Définition
Disponibilité	Part des exécutions réussies (`OK` ou `WARNING`) parmi les exécutions évaluées sur une période ; les exécutions `UNKNOWN` en sont exclues. Voir chapitre 17
DNS	Système de résolution des noms de domaine en adresses IP
Données de performance (*perfdata*)	Mesures chiffrées renvoyées par un plugin Nagios après le caractère `|` de sa sortie
Drainage	Phase qui ouvre l'activation d'une mise à jour : la sonde ne démarre plus aucune exécution et laisse celles en cours se terminer, dans la limite du délai `updates.drainTimeout` (5 minutes par défaut, 30 minutes au maximum). Au-delà, elles sont interrompues en `UNKNOWN` avec le type d'erreur `maintenance` . Voir chapitre 20
Dynatrace	Plateforme d'observabilité. Elle reçoit les données de la sonde via son API OTLP. Voir chapitre 18
EAP-TLS	Méthode d'authentification 802.1X fondée sur des certificats : la sonde présente un certificat client et vérifie celui du serveur RADIUS. Les échanges avec le commutateur passent par des trames EAPOL, au niveau de la couche liaison (aucun port IP). Seule méthode 802.1X prise en charge
Édition	Niveau fonctionnel de la sonde (Essential, Professional, Enterprise), déterminé par le certificat d'édition inscrit sur la carte (voir **Certificat d'édition**). L'image système est commune à toutes les éditions
Empreinte (SHA-256)	Valeur calculée qui identifie de manière unique un certificat ou un fichier. On la compare pour vérifier l'authenticité du certificat de la sonde
Encart d'identification	Document qui accompagne la carte microSD et indique l'édition acquise, le numéro de série du produit et le code d'installation. À conserver en lieu sûr : il est nécessaire à la mise en service et à la récupération du compte administrateur. Voir chapitre 5
Endpoint	Adresse (URL) d'un service auquel on envoie des requêtes, par exemple l'endpoint OTLP d'une plateforme
Enregistreur Chrome DevTools (*Recorder*)	Outil intégré à Chrome qui enregistre un parcours réalisé à la main et l'exporte (JSON, script Puppeteer). Aide à la conception des scénarios ; successeur de l'extension Resurrectio. Voir chapitre 14
Environnement isolé	Contexte d'exécution séparé (compte système distinct, limites de ressources) dans lequel la sonde exécute les scénarios et les plugins, sans accès à sa base ni aux secrets non déclarés
ETag	Identifiant de version d'une ressource de l'API. Il est renvoyé dans l'en-tête `If-Match` pour éviter d'écraser une modification concurrente
Étape	Subdivision nommée et mesurée d'un scénario ou d'une suite de requêtes API. Elle permet de localiser un échec
État	Résultat synthétique d'un contrôle : `OK` , `WARNING` , `CRITICAL` , `UNKNOWN` ou `STALE` . Voir annexe I
État de maintenance	État de la sonde pendant l'activation d'une mise à jour, du début du drainage jusqu'au retour en service après le redémarrage (événement `maintenance.ended`) : `GET /health` répond `MAINTENANCE` (code 503), `check_sva` renvoie 3 avec le texte `MAINTENANCE` , la métrique `sva.agent.maintenance` vaut 1. Il ne suffit pas à neutraliser les alertes, car la sonde est ensuite injoignable pendant son redémarrage. À ne pas confondre avec le justificatif de maintenance. Voir chapitres 18 et 20
Exécution	Occurrence de lancement d'un contrôle, avec son état, sa durée, ses étapes et ses éventuels artefacts. Identifiée par un `runId`
Exécution d'essai (*dry run*)	Exécution déclenchée manuellement qui n'enregistre pas d'état et n'exporte rien. Utilisée pour tester un contrôle
Faux positif, faux négatif	Un faux positif est une alerte émise alors que la ressource fonctionne ; un faux négatif, l'absence d'alerte alors qu'elle est défaillante. Voir chapitre 17
Fenêtre d'activité	Plage horaire pendant laquelle un plan d'exécution est actif (`activeWindows`), par exemple les heures ouvrées

Terme	Définition
Fenêtre de maintenance	Plage récurrente (jours, heure de début, durée), définie par le client dans la section `updates` (`windows`, 7 au plus), pendant laquelle une mise à jour déjà installée peut être activée, c'est-à-dire la sonde redémarrée sur la nouvelle version. Aucune fenêtre n'est définie par défaut ; sans fenêtre, aucune activation automatique n'a lieu. Voir chapitre 20
Fichier de provisionnement	Fichier `provision.yaml` déposé sur la partition de démarrage de la microSD avant le premier démarrage, pour configurer la sonde sans assistant. Voir chapitre 6
Gigue (*jitter*)	Décalage aléatoire ajouté au démarrage des exécutions pour éviter qu'elles ne partent toutes au même instant
Grafana	Outil de visualisation de données, souvent associé à Prometheus, Mimir, Tempo et Loki
Headless	Mode de fonctionnement d'un navigateur sans fenêtre affichée. Les pages sont chargées et exécutées normalement
Headless shell	Variante allégée de Chromium, dédiée au fonctionnement sans fenêtre. Elle n'est pas utilisée par la sonde, car elle n'est pas distribuée pour l'architecture du Raspberry Pi 5 : la sonde emploie le mode headless du navigateur complet. Voir chapitre 13
High Endurance	Catégorie de cartes microSD conçues pour des écritures continues (vidéosurveillance). C'est le type de la carte fournie par l'éditeur. Utilisez exclusivement cette carte : toute carte de remplacement s'obtient selon le § 20.6.4, car une carte achetée ailleurs démarrerait en mode restreint. Voir chapitre 20
Histogramme	Type de métrique qui répartit des mesures (durées) dans des intervalles, et permet de calculer des centiles
HTTP / HTTPS	Protocole du Web ; HTTPS en est la version chiffrée par TLS
ICMP	Protocole de contrôle du réseau, utilisé notamment par la commande `ping`. Son équivalent en IPv6, ICMPv6, assure aussi la découverte des voisins (NDP)
Iframe	Cadre qui affiche une page web à l'intérieur d'une autre. Un scénario doit cibler explicitement l'iframe pour agir sur son contenu
Image système	Contenu préinstallé de la carte microSD : système d'exploitation, logiciels de la sonde et configuration par défaut. Une image est propre à une famille de cartes. Les mises à jour sont livrées sous forme d'image signée (*bundle*), vérifiée avant installation (voir **A/B**)
Indicateur métier	Valeur fonctionnelle mesurée par la sonde (nombre de commandes du jour, par exemple), exportée comme métrique `sva.indicator.value`
Interface d'administration	Interface web servie par la sonde en HTTPS, pour la configurer et consulter ses résultats
IPv6	Version 6 du protocole IP. La sonde fonctionne en double pile : IPv4 obligatoire, IPv6 activable (désactivé par défaut), par autoconfiguration (SLAAC) ou adresse statique. Le paramètre `ipFamily` d'un contrôle choisit la famille d'adresses. Voir chapitre 7
JDBC	Interface standard de connexion aux bases de données depuis Java, utilisée par le moteur principal pour les contrôles `database`
Jeton d'API	Chaîne secrète qui authentifie un outil ou un utilisateur auprès de l'API, avec un rôle déterminé. Elle n'est affichée qu'une fois, à sa création
Journal d'audit	Registre des opérations d'administration : connexions, modifications, téléversements, mises à jour, activations de SSH, récupérations du compte administrateur
JSON	Format texte de données structurées, utilisé par l'API et les définitions de contrôles
JSON Merge Patch	Format de modification partielle d'un document JSON (RFC 7386), utilisé par les requêtes `PATCH` de l'API

Terme	Définition
JSONPath	Langage d'expression permettant de désigner une valeur dans un document JSON, par exemple `$.available`
Justificatif de maintenance	Fichier signé par l'éditeur, distinct du certificat d'édition, importé hors ligne lorsque le client souscrit le contrat de maintenance facultatif. Il porte une date de fin et donne accès aux nouvelles versions fonctionnelles. Son échéance n'affecte aucune fonction acquise ni l'accès aux correctifs de sécurité. Voir chapitre 20
Lanceur local	Outil pédagogique fourni avec les exemples, qui exécute un scénario scripté ou déclaratif sur un poste de travail en reproduisant le contrat du moteur de scénarios. Il est distinct du produit. Voir chapitres 14 et 15
Licence	Droit d'utilisation perpétuel du logiciel intégré, attaché à la carte microSD livrée. Il se matérialise par le certificat d'édition : aucune clé n'est à saisir, aucun renouvellement n'est nécessaire. À ne pas confondre avec le justificatif de maintenance
Locator	Objet de l'API Puppeteer qui désigne un élément de la page et attend automatiquement qu'il soit utilisable avant d'agir
Log (journal)	Enregistrement horodaté d'un événement. Les logs d'exécution de la sonde peuvent être exportés via OpenTelemetry. Voir chapitre 18
MAB (*MAC Authentication Bypass*)	Exception accordée par le commutateur à un équipement d'après son adresse MAC, sans authentification 802.1X. Alternative à EAP-TLS, notamment pour la première mise en service. Voir chapitre 7
mDNS	Protocole qui permet de résoudre des noms en `.local` sur un réseau local sans serveur DNS
Métrique	Mesure numérique horodatée (durée, état, nombre d'exécutions) transmise aux plateformes d'observabilité
Mode réduit (SSD)	Fonctionnement d'une sonde dont le SSD NVMe activé est absent ou défaillant : les contrôles continuent, l'historique local est conservé sur la microSD et l'état de santé `storage` passe en `CRITICAL`. À ne pas prolonger, car l'usure de la microSD s'accélère. Distinct du **mode restreint**. Voir chapitre 20, § 20.3.4
Mode restreint	État de la sonde lorsque le certificat d'édition est absent, invalide ou ne correspond pas à la carte, ou lorsque la période d'une carte de démonstration est terminée : aucun contrôle n'est exécuté, mais la consultation, l'export de la configuration et le diagnostic restent possibles. Voir chapitre 6, § 6.5
Moteur de scénarios	Composant de la sonde fondé sur Node.js, Puppeteer et Chromium, qui exécute les scénarios navigateur dans l'environnement isolé
Moteur principal	Composant Java de la sonde : planification, API, interface d'administration, contrôles techniques et applicatifs, stockage, export
mTLS	Authentification TLS mutuelle : le client présente aussi un certificat. Prévue pour la console centrale
MTTD, MTTR	Temps moyen de détection (*Mean Time To Detect*) et temps moyen de rétablissement (*Mean Time To Restore*) d'un incident. La supervision synthétique vise d'abord à réduire le MTTD. Voir chapitre 2
Nagios	Logiciel de supervision de référence, dont le format de plugins (codes retour 0 à 3) est largement repris
Node.js	Environnement d'exécution JavaScript utilisé par le moteur de scénarios
Nouvelle tentative (*retry*)	Réexécution automatique d'un contrôle après un échec, avant de confirmer l'état `CRITICAL`
NTP	Protocole de synchronisation de l'heure, indispensable à la sonde
OAuth 2.0	Protocole d'autorisation qui délivre des jetons d'accès aux API. Le flux *client credentials* permet à un compte technique d'obtenir un jeton. Voir chapitre 11
Observabilité	Capacité à comprendre l'état d'un système à partir des données qu'il émet : métriques, traces et logs

Terme	Définition
Onduleur	Alimentation secourue qui protège la sonde des coupures et micro-coupures électriques. Recommandé. Voir chapitre 20
OpenAPI	Format standard de description d'une API REST. La sonde publie la description de son API à l'adresse `/api/v1/openapi.json`. Voir chapitre 9
OpenTelemetry	Standard ouvert de production et de transport des métriques, traces et logs
OTLP	Protocole de transport d'OpenTelemetry, en HTTP ou gRPC
Partition de démarrage	Partition de la microSD au format FAT32, étiquetée `SVA-BOOT`, lisible depuis tout poste de travail. On y dépose `provision.yaml` (et, le cas échéant, les fichiers 802.1X qu'il désigne), `recovery.txt` et `factory-reset.txt` ; la sonde y écrit `setup-code.txt`, `provision-result.txt` et `factory-reset-challenge.txt`. Avec le mécanisme A/B, d'autres partitions de la carte peuvent apparaître sur le poste : elles ne doivent pas être modifiées. La partition `SVA-BOOT` reste sur la microSD, même lorsqu'un SSD est installé
PEM	Format texte d'un certificat ou d'une clé, encodé en Base64 entre des lignes `-----BEGIN …-----` et `-----END …-----`. Format attendu pour les certificats d'autorité et pour le certificat 802.1X signé importé sur la sonde. Voir chapitre 7
Période de support	Période pendant laquelle l'éditeur fournit les correctifs de sécurité d'une version majeure, sans contrat de maintenance : au moins 5 ans après sa mise sur le marché. Sa date de fin (mois et année) est indiquée à l'achat et exposée par l'API (`supportEndsAt`). Au-delà, la sonde continue de fonctionner, sans nouveau correctif. Voir chapitre 20
PKCS#12	Format de fichier protégé par mot de passe qui regroupe une clé privée et son certificat (extensions `.p12` ou `.pfx`). Accepté pour l'import d'un identifiant 802.1X. Voir chapitre 7
Plan d'exécution (*playlist*)	Liste ordonnée de contrôles exécutés en séquence selon une planification commune. Voir chapitre 12
Plugin Nagios	Programme respectant la convention Nagios (code retour, sortie texte, données de performance), exécuté par un contrôle `nagios`
PoE (*Power over Ethernet*)	Alimentation électrique par le câble réseau. Hors du périmètre de prise en charge : la sonde s'alimente par son port USB-C, et non par le module d'extension PoE+ officiel du Raspberry Pi 5
Préavis	Délai minimal entre l'annonce d'une activation de mise à jour et son début (`updates.notice`, 24 heures par défaut, 7 jours au maximum). Une activation qui ne peut respecter le préavis passe à la fenêtre suivante. Voir chapitre 20
Procédure stockée	Programme enregistré dans une base de données, exécutable par un contrôle `database` (champ `procedure`). Voir chapitre 11
Profil éditeur	Fichier signé par l'éditeur, intégré à l'image et actualisé par le canal de mises à jour `security`, qui contient les adresses de l'éditeur utilisées par la sonde (dépôt de mises à jour, documentation, support, téléchargements), dans le domaine `incitypark.fr`. Aucune adresse de l'éditeur n'est inscrite dans le logiciel. Voir chapitre 20
Prometheus	Système libre de collecte et de stockage de métriques. Il interroge la sonde sur `/metrics` ou reçoit ses données via OTLP
PromQL	Langage de requête de Prometheus, utilisé pour les tableaux de bord et les règles d'alerte. Voir chapitre 18
Proxy	Serveur intermédiaire par lequel passent les connexions sortantes vers Internet
Puppeteer	Bibliothèque JavaScript qui pilote Chromium par programme. Elle est utilisée par les scénarios scriptés
RADIUS	Protocole et serveur d'authentification centralisée. Dans un déploiement 802.1X, le commutateur lui transmet l'authentification de la sonde ; la sonde vérifie le certificat du serveur RADIUS. Voir chapitre 7

Terme	Définition
Récepteur OTLP	Composant qui reçoit des données OTLP : collecteur OpenTelemetry, Agent Datadog, Prometheus lancé avec `--web.enable-otlp-receiver`. Voir chapitre 18
Réinitialisation usine	Retour de la sonde à son état de livraison : configuration, comptes, secrets et historique sont effacés ; le certificat d'édition est conservé et le code d'installation imprimé redevient valide. Sans accès administrateur, elle se déclenche par le fichier `factory-reset.txt`, avec une confirmation en deux temps par un code aléatoire. Voir chapitre 20
Rétention	Durée de conservation des données sur la sonde (exécutions, agrégats, artefacts, audit)
Rôle	Ensemble de droits attribué à un compte ou un jeton : `admin`, `operator`, `viewer`, `metrics`
RUM (*Real User Monitoring*)	Mesure de l'expérience des utilisateurs réels, collectée dans leur navigateur. Complémentaire de la supervision synthétique. Voir chapitre 2
SaaS	Logiciel fourni sous forme de service hébergé par son éditeur (Datadog, Dynatrace, Grafana Cloud, par exemple)
Scénario	Contrôle de type `browser` qui reproduit un parcours utilisateur dans un navigateur
Scénario déclaratif	Scénario décrit par une liste d'étapes JSON, sans code
Scénario scripté	Scénario écrit en JavaScript avec Puppeteer, réservé au rôle `admin`
Secret	Valeur sensible (mot de passe, jeton, clé) stockée chiffrée par la sonde, jamais relisible, transmise uniquement aux contrôles qui la déclarent
Sélecteur	Expression qui désigne un élément d'une page web (sélecteur CSS, texte, rôle d'accessibilité, XPath)
Server-Sent Events (SSE)	Flux d'événements unidirectionnel sur HTTP. Utilisé par `GET /api/v1/events` pour suivre les exécutions en temps réel. Voir chapitres 9 et 17
Seuil de performance	Durée au-delà de laquelle une exécution réussie passe en `WARNING` (`warningDuration`) ou en `CRITICAL` (`criticalDuration`)
SLA, SLO	Engagement de niveau de service (*Service Level Agreement*) et objectif de niveau de service (*Service Level Objective*), par exemple un taux de réussite d'un parcours sur un mois. Voir chapitre 2
SLAAC (*Stateless Address Autoconfiguration*)	Autoconfiguration d'une adresse IPv6 à partir des annonces du routeur, sans serveur DHCP. Mode IPv6 par défaut de la sonde lorsque l'IPv6 est activé. Voir chapitre 7
SOAP	Protocole d'échange de messages XML utilisé par certains web services. Il se supervise par un contrôle `http` en `POST` XML. Voir chapitre 11
Sonde	Appareil de supervision constitué d'un Raspberry Pi 5 et du logiciel préinstallé sur la carte microSD
Sous-tension	Alimentation électrique insuffisante, détectée par le Raspberry Pi. Cause fréquente d'instabilité et de corruption
Span	Élément d'une trace qui représente une opération et sa durée (une étape de scénario, par exemple)
SQLite	Base de données embarquée utilisée par la sonde pour sa configuration et ses résultats
SSD NVMe	Disque à mémoire flash raccordé en PCIe. Sur la sonde, un SSD M.2 2230 ou 2242 monté sur la carte M.2 HAT+ officielle peut, en option, accueillir les données (historique, artefacts, journaux). Le système et le démarrage restent sur la microSD. Voir chapitres 4, 5 et 20
SSH	Protocole d'accès à distance en ligne de commande. Désactivé par défaut sur la sonde ; un administrateur peut l'activer pour une durée limitée (4 heures par défaut, 72 heures au maximum) ou le verrouiller définitivement. Voir chapitre 7
`STALE`	État d'un contrôle actif dont aucune exécution ne s'est terminée depuis plus de deux intervalles de planification. Il qualifie le contrôle, et non une exécution. Voir chapitre 8 et annexe I

Terme	Définition
Supervision applicative	Vérification du bon fonctionnement des applications du point de vue de leurs utilisateurs et des règles métier, au-delà de la disponibilité des serveurs
Supervision synthétique	Supervision par simulation de requêtes ou de parcours utilisateurs, exécutés à intervalles réguliers depuis un point d'observation. Lorsqu'elle s'appuie sur plusieurs sondes installées dans des environnements différents (réseau interne, sites distants, clients, partenaires, régions), on parle de supervision synthétique distribuée. Voir chapitre 2, § 2.6.1
`sva` (préfixe)	Préfixe technique des composants et des identifiants du logiciel de la sonde, issu de sa désignation technique interne « SVA ». On le retrouve dans le nom de la partition de démarrage de la microSD (`SVA-BOOT`), des services (`sva-agent` , `sva-runner`), du nom d'hôte par défaut (`sva-XXXXXX`), du plugin Nagios (`check_sva`), des métriques (`sva.*` , écrites `sva_*` au format Prometheus) et du répertoire de travail utilisé par les exemples sur le poste d'administration (`$HOME/.sva` , chapitre 9, § 9.3.3). Ces identifiants sont stables et indépendants du nom commercial SYNOVEX (chapitre 1)
SYNOVEX	Nom commercial de la gamme de sondes décrite dans cette documentation (éditions SYNOVEX Essential, Professional et Enterprise). Les identifiants techniques (`sva` , `sva-agent` , `sva-runner` , `check_sva`) proviennent de la désignation technique interne « SVA » et en sont indépendants
TCP	Protocole de transport orienté connexion, utilisé par la plupart des services réseau
Timeout (délai maximal)	Durée au-delà de laquelle une exécution ou une attente est interrompue et considérée en échec
TLS	Protocole de chiffrement des communications, utilisé par HTTPS
Trace	Représentation d'une exécution sous forme d'arbre de spans, transmise via OpenTelemetry
tryboot	Mécanisme du chargeur d'amorçage du Raspberry Pi qui démarre une seule fois sur une partition donnée. La sonde l'utilise pour essayer un nouvel emplacement système avant de le valider (voir **A/B**)
Type d'erreur	Catégorie normalisée de la cause d'un échec (`timeout` , `dns_error` …). Voir annexe H
ULID	Format d'identifiant unique et triable chronologiquement, utilisé pour les exécutions (`runId`)
UTC	Temps universel coordonné. La sonde horodate ses données en UTC ; l'affichage peut utiliser le fuseau horaire local
Viewport (fenêtre d'affichage)	Dimensions de la fenêtre simulée par le navigateur d'un scénario. Voir chapitre 14
VLAN	Réseau local virtuel qui segmente logiquement un réseau physique. Un VLAN dédié à la supervision est recommandé. La sonde n'étiquette pas ses trames : elle se raccorde à un port d'accès configuré dans ce VLAN. Voir chapitre 7
W3C Trace Context (`traceparent`)	Norme de propagation du contexte de trace entre systèmes. La sonde peut, en option, l'injecter dans ses requêtes. Voir chapitre 18
Watchdog (chien de garde)	Mécanisme qui redémarre automatiquement un service ou l'appareil s'il cesse de répondre
XPath	Langage de désignation d'éléments dans un document XML ou HTML. Utilisable dans les assertions et, en dernier recours, comme sélecteur
YAML	Format texte de configuration fondé sur l'indentation, utilisé par le fichier de provisionnement

21.2 Annexe B — Questions fréquentes

Fonctionnement général

La sonde a-t-elle besoin d'Internet pour fonctionner ? Non. Elle exécute ses contrôles, conserve ses résultats et sert son interface d'administration sans aucune connexion à Internet. Un accès sortant n'est nécessaire que pour les fonctions qui l'exigent : mises à jour en ligne, envoi de données vers une plateforme SaaS (Datadog, Dynatrace, Grafana Cloud), supervision de sites publics. Voir chapitre 3.

La sonde dépend-elle d'un serveur de l'éditeur ? Non. Elle est autonome. L'indisponibilité d'une plateforme externe ou de la future console centrale n'interrompt pas ses contrôles.

Quelles données sont envoyées à l'éditeur ? Aucune donnée de supervision n'est envoyée à l'éditeur. Les résultats ne partent que vers les destinations d'export que vous configurez. Les seuls échanges avec l'éditeur concernent la vérification et le téléchargement des mises à jour, si vous les activez. La sonde n'envoie aucune télémétrie à l'éditeur : ni statistiques d'usage, ni données de diagnostic automatiques. L'édition est vérifiée localement : aucune vérification de licence en ligne n'a lieu. Une archive de diagnostic n'est transmise au support que si vous la lui envoyez vous-même.

Le droit d'utilisation de la sonde expire-t-il ? Non. Le droit d'utilisation du logiciel intégré est perpétuel et attaché à la carte. Il ne nécessite ni abonnement ni renouvellement. Un contrat de maintenance facultatif peut être souscrit séparément ; son échéance n'interrompt pas le fonctionnement de la sonde. Seules les cartes de démonstration, prêtées pour une évaluation, ont une durée limitée.

Que se passe-t-il à la fin du contrat de maintenance ? Rien ne change dans le fonctionnement de la sonde : toutes les fonctions acquises restent disponibles et les correctifs de sécurité restent accessibles pendant la période de support. Le contrat de maintenance, facultatif, se matérialise par un justificatif de maintenance signé, importé hors ligne sur la sonde. À son échéance, seul l'accès aux nouvelles versions fonctionnelles et à l'assistance prévue par le contrat prend fin. Voir chapitre 20.

Pendant combien de temps les correctifs de sécurité sont-ils fournis ? Pendant la période de support : au moins 5 ans après la mise sur le marché de chaque version majeure, durée allongée si la durée d'utilisation prévisible du produit ou la réglementation l'exige. Ces correctifs ne nécessitent aucun contrat de maintenance. La date de fin (mois et année) vous est indiquée à l'achat ; elle figure aussi dans l'état de la sonde (`GET /api/v1/system`, champ `supportEndsAt`). Après cette date, la sonde continue de fonctionner, sans nouveau correctif, et l'état de santé `support` passe en `WARNING`. Voir chapitre 20.

Dans quel délai les failles de Chromium sont-elles corrigées ? L'éditeur s'est fixé des objectifs, publiés mais non contractuels : 14 jours pour une vulnérabilité critique ou activement exploitée, 30 jours pour une vulnérabilité élevée, à compter de la disponibilité du correctif en amont. Les engagements contractuels éventuels relèvent de votre contrat de maintenance, selon ses conditions. Voir chapitre 20.

Puis-je changer de Raspberry Pi 5 ? Oui. L'édition est attachée à la carte microSD, et non au Raspberry Pi. Vous pouvez déplacer la carte sur un autre Raspberry Pi 5 compatible, de capacité mémoire suffisante pour l'édition. En revanche, une copie du système sur une autre carte démarre en mode restreint. Voir chapitre 6, § 6.5.

Que se passe-t-il en cas de coupure de courant ? La sonde redémarre seule au retour du courant et reprend ses contrôles. Elle vérifie l'intégrité de sa base et la restaure si nécessaire. Les contrôles prévus pendant la coupure ne sont pas rattrapés. Un onduleur est recommandé. Voir chapitre 20.

Que se passe-t-il si la plateforme d'observabilité est indisponible ? Les contrôles continuent et l'historique reste consultable sur la sonde. Les envois sont retentés pendant une durée limitée. Au-delà, les métriques de la période ne sont pas transmises. Voir chapitre 18.

Combien de contrôles une sonde peut-elle exécuter ? Aucun quota technique ne limite le nombre de contrôles, de scénarios ou de plans, quelle que soit l'édition. La capacité réelle dépend du type et de la fréquence des contrôles et de la mémoire du Raspberry Pi 5. Plusieurs dizaines de contrôles techniques par minute sont envisageables. Les scénarios navigateur, beaucoup plus coûteux, sont exécutés un à la fois par défaut. La méthode de dimensionnement d'une sonde selon sa charge est décrite au chapitre 12, § 12.6.

Quelle fréquence de contrôle choisir ? Une à cinq minutes pour les contrôles techniques critiques, cinq à quinze minutes pour les parcours navigateur, selon le délai de détection visé et la charge acceptable pour l'application. Voir chapitre 8.

Matériel et réseau

Puis-je utiliser un autre modèle que le Raspberry Pi 5 ? Le Raspberry Pi 5 est la seule carte pour laquelle SYNOVEX est conçu, documenté et pris en charge. Son architecture logicielle peut permettre une utilisation sur d'autres ordinateurs monocartes, notamment certains modèles Orange Pi. Toutefois, ces plateformes n'ont pas fait l'objet d'une validation officielle et leur fonctionnement n'est pas garanti. La carte microSD livrée contient une image système propre au Raspberry Pi 5 : elle ne peut pas être utilisée sur une autre carte (chapitre 4, § 4.1.5).

La sonde fonctionne-t-elle en Wi-Fi ? Non : la sonde fonctionne uniquement en Ethernet. Ce choix garantit la stabilité des mesures et simplifie la sécurité. Une connexion Wi-Fi est à l'étude pour une version ultérieure (§ 1.4.4).

Puis-je alimenter la sonde par le câble réseau (PoE) ? Le Raspberry Pi 5 nécessite un module d'extension pour être alimenté en PoE. Cette configuration n'est pas prise en charge. Utilisez l'alimentation USB-C recommandée.

La sonde prend-elle en charge 802.1X, IPv6 et les VLAN ?

- **802.1X** : oui, avec la méthode EAP-TLS (certificat client). La clé et la demande de certificat peuvent être générées sur la sonde, ou un fichier PKCS#12 importé. Pour un port contrôlé dès le premier démarrage, l'identifiant 802.1X se dépose avec le fichier de provisionnement. Sinon, réalisez la première mise en service sur un port non contrôlé ou avec une exception par adresse MAC (MAB), puis activez 802.1X. La méthode PEAP n'est pas prise en charge.
- **IPv6** : oui, en double pile. IPv4 reste obligatoire ; IPv6 est activable (désactivé par défaut), en SLAAC ou en adresse statique, et les contrôles peuvent viser des cibles IPv6.
- **VLAN** : la sonde n'étiquette pas ses trames. Raccordez-la à un port d'accès configuré dans le VLAN de supervision.

Voir chapitre 7.

La sonde peut-elle utiliser un SSD ? Oui, en option, dans toutes les éditions. Un SSD NVMe M.2 2230 ou 2242 de la liste des modèles pris en charge (notes de version), monté sur la carte M.2 HAT+ officielle de Raspberry Pi, accueille alors les données de la sonde : historique, agrégats, captures, file d'export et journaux persistants. Le démarrage, le système, la configuration, les secrets et le certificat d'édition restent sur la microSD, qui demeure le support principal et suffit à un usage standard. Le SSD est surtout utile à l'édition Enterprise. Son activation formate le SSD et redémarre la sonde. Voir chapitres 4, 5 et 20.

Quels ports dois-je ouvrir ? En entrant, le port TCP 443 depuis le réseau d'administration (et depuis Prometheus, Nagios ou Centreon s'ils l'interrogent). En sortant : DNS, NTP, les ressources supervisées, et selon vos choix les destinations d'export et le dépôt de mises à jour. Si l'IPv6 est activé, les mêmes règles s'appliquent en IPv6 et ICMPv6 doit être autorisé pour la découverte des voisins. L'authentification 802.1X n'utilise aucun port IP. Voir annexe E.

La sonde peut-elle passer par notre proxy ? Oui, pour les flux HTTP et HTTPS sortants, avec authentification Basic facultative et une liste d'exclusions. Voir chapitre 7.

Comment superviser plusieurs sites ? Installez une sonde par site. Chaque sonde porte un nom de site (`sva.site`) qui permet de comparer, dans vos plateformes, la vision d'une même application depuis plusieurs emplacements. La console centrale multi-sondes est une évolution prévue.

Contrôles et scénarios

Faut-il savoir programmer pour créer des contrôles ? Non pour les contrôles techniques, applicatifs et les scénarios déclaratifs, qui se configurent en JSON ou depuis l'interface. Les scénarios scriptés requièrent des notions de JavaScript. Voir chapitres 13 à 16.

Puis-je réutiliser mes plugins Nagios ? Oui. Les plugins standard du paquet `monitoring-plugins` sont fournis, à l'exception des plugins privilégiés (`check_icmp`, `check_dhcp`) ; la liste exacte des plugins intégrés figure dans les notes de version. Les plugins personnalisés sont pris en charge à partir de l'édition Professional, sous réserve qu'ils ne nécessitent pas de privilèges élevés. Voir chapitre 10.

Puis-je réutiliser mes anciens scénarios CasperJS ? Pas directement. CasperJS n'est plus maintenu. Les scénarios doivent être réécrits avec Puppeteer, en conservant leur intention fonctionnelle. L'annexe J et les chapitres 13 à 16 facilitent cette conversion.

Faut-il superviser la production ou un environnement de test ? Les deux sont possibles. En production, limitez les scénarios à des parcours sans effet (consultation, authentification, recherche) ou utilisez des comptes et des produits de test prévus par l'application. Ne déclenchez jamais de commande ou de paiement réel. Voir chapitre 15.

Comment gérer une application protégée par une authentification multifacteur ? La sonde ne contourne pas l'authentification multifacteur. Demandez un compte de test exempté dans un environnement prévu à cet effet, ou une méthode d'authentification adaptée aux comptes techniques, selon votre politique de sécurité. Voir chapitre 16.

Mes scénarios fonctionnent sur mon poste mais pas sur la sonde. Pourquoi ? La sonde démarre chaque scénario avec un navigateur vierge, depuis son propre réseau, avec sa propre résolution DNS et sans votre session. Voir chapitre 19, § 19.7.

Les scénarios peuvent-ils fausser les statistiques de mon site ? Oui, comme tout visiteur. Excluez l'adresse IP de la sonde de vos outils de mesure d'audience, ou bloquez ces outils dans le scénario (`blockDomains`).

Sécurité et conformité

Où sont stockés les mots de passe des comptes de test ? Sur la sonde, chiffrés. Ils ne peuvent jamais être relus par l'API ou l'interface, et ne sont transmis qu'aux contrôles qui les déclarent. Voir chapitre 9.

Existe-t-il un mot de passe par défaut ? Non. Le compte administrateur est créé lors de la mise en service.

J'ai perdu le mot de passe du dernier administrateur. Que faire ? Si un autre compte `admin` existe, il peut redéfinir le mot de passe. Sinon, utilisez la procédure de récupération, qui conserve la configuration. Elle exige un accès physique à la carte microSD, le code d'installation imprimé et le **code de récupération** affiché à la fin de la mise en service. Déposez le fichier `recovery.txt` sur la partition de démarrage, puis redémarrez la sonde : une page de récupération est ouverte pendant 30 minutes, avec 3 essais au plus. Vous y définissez un nouveau mot de passe pour un compte `admin` existant ; ses sessions et jetons sont révoqués. Le code de récupération utilisé est alors remplacé par un nouveau code, à conserver de nouveau. Sans code de récupération, seule la réinitialisation usine reste possible : elle se confirme en deux temps par un code aléatoire, valable 24 heures, que la sonde écrit sur la partition de démarrage. Voir chapitre 20.

Le support de l'éditeur peut-il accéder à la sonde à distance ? Non. L'éditeur ne dispose d'aucun accès permanent. Le support travaille à partir de l'archive de diagnostic que vous lui transmettez. Si une intervention en ligne de commande est nécessaire, seul un administrateur de la sonde peut activer SSH, pour une durée limitée (4 heures par défaut, 72 heures au maximum) ; l'activation, les connexions et la désactivation sont journalisées. Vous pouvez aussi verrouiller SSH définitivement : ce choix n'est réversible que par une réinitialisation usine et empêche toute intervention du support par SSH. Voir chapitre 7.

Quelles sont les implications au regard du RGPD ? La sonde peut collecter des données personnelles si les pages supervisées en affichent (captures d'écran) ou si les comptes de test en contiennent. Utilisez des comptes de test sans données réelles, limitez la rétention des artefacts et intégrez la sonde à votre registre des traitements si nécessaire. Le client reste responsable des données traitées par ses scénarios.

Puis-je exposer la sonde sur Internet ? Non. L'interface d'administration et l'API sont destinées au réseau où la sonde est installée. Pour une administration à distance, utilisez un moyen d'accès sécurisé à ce réseau (VPN d'administration, bastion), celui de votre organisation ou celui convenu avec l'exploitant du réseau d'accueil. Cette règle ne limite pas l'emplacement de la sonde : elle peut être installée sur un site distant, chez un client ou chez un partenaire, sans être exposée sur Internet.

Qui peut exécuter du code sur la sonde ? Seuls les comptes de rôle `admin` peuvent déposer des scénarios scriptés et des plugins personnalisés. Ce code s'exécute dans l'environnement isolé.

Exploitation

Comment être alerté si la sonde elle-même tombe en panne ? Configurez dans votre plateforme une alerte sur l'absence de la métrique `sva.agent.up`, ou un service Nagios ou Centreon sur `check_sva`. Voir chapitres 18 et 20.

Les mises à jour sont-elles automatiques ? Vous choisissez, séparément pour les correctifs de sécurité (automatiques par défaut) et pour les versions fonctionnelles (manuelles par défaut). Le téléchargement et l'installation ont lieu sans interrompre les contrôles ; seul le redémarrage sur la nouvelle version (activation) attend une fenêtre de maintenance que vous définissez, ou l'action d'un administrateur. Aucune fenêtre n'est définie par défaut. Chaque mise à jour est une image système signée, vérifiée avant installation, puis installée sur l'emplacement système inactif (mécanisme A/B). La sonde redémarre une fois sur la nouvelle version, puis dispose de 10 minutes pour réussir son autotest ; à défaut, elle revient automatiquement à la version précédente. Ces 10 minutes sont le délai accordé à l'autotest, et non une durée garantie de retour en fonctionnement : si le système se bloque, le retour suppose un redémarrage effectif, par un chien de garde ou une remise sous tension (chapitre 20, § 20.4.5). Une sonde sans accès au dépôt de l'éditeur se met à jour avec le même fichier signé, importé hors ligne. Voir chapitre 20.

Les mises à jour peuvent-elles redémarrer la sonde à un moment gênant ? Pas avec les réglages par défaut. Une mise à jour ordinaire n'est activée que dans l'une de vos fenêtres de maintenance, annoncée au moins 24 heures à l'avance par défaut (préavis). Un correctif critique, qui corrige une vulnérabilité activement exploitée, est activé par défaut dans la première fenêtre qui commence au moins 15 minutes après son annonce. Sans fenêtre, aucun de ces redémarrages n'a lieu : un administrateur active la mise à jour, immédiatement ou à l'heure de son choix. Seul le mode `asap`, que vous choisissez explicitement, active un correctif critique hors de vos fenêtres, 15 minutes après son annonce augmentées du décalage de la sonde, même si aucune fenêtre n'est définie. Vous pouvez reporter une activation (30 jours au plus, 7 jours pour un correctif critique). Aucun redémarrage immédiat n'est jamais imposé. La matrice complète figure au chapitre 20, § 20.4.2. Le règlement européen sur la cyberrésilience (CRA) n'impose aucun horaire. Voir chapitre 20.

Comment éviter les fausses alertes pendant une mise à jour ? Configurez dans vos plateformes des plages de neutralisation alignées sur les fenêtres de maintenance de la sonde (Alertmanager, Datadog, Dynatrace, Centreon, Nagios). L'état de maintenance (`/health` à `MAINTENANCE`, métrique `sva.agent.maintenance`) ne suffit pas, car la sonde est injoignable pendant son redémarrage. Voir chapitre 18.

Faut-il plusieurs sondes pour une supervision sans interruption ? Oui, pour les environnements critiques. Une sonde unique ne supervise pas pendant son propre redémarrage. Déployez au moins deux sondes par site, exécutant les mêmes contrôles, avec des fenêtres de maintenance qui ne se chevauchent pas, et fondez vos alertes sur l'ensemble des sondes du site. Voir chapitres 18 et 20.

Comment arrêter ou redémarrer la sonde à distance ? Depuis l'interface d'administration, ou par l'API avec un compte `admin` : `POST /api/v1/system/shutdown` pour l'arrêt, `POST /api/v1/system/reboot` pour le redémarrage. Après un arrêt, un redémarrage nécessite une intervention physique (débrancher puis rebrancher

l'alimentation, ou appui sur le bouton d'alimentation). Voir chapitre 20, § 20.2.

Comment configurer mes outils pour faire confiance au certificat de la sonde ? Récupérez le certificat au format PEM (`GET /api/v1/system/certificate` ou export depuis le navigateur), vérifiez son empreinte avec celle du fichier `setup-code.txt` ou de l'écran HDMI, puis déclarez-le dans vos outils (`--cacert` pour `curl`, `tls_config.ca_file` pour Prometheus). Le plus simple reste d'installer sur la sonde un certificat émis par l'autorité de certification de votre entreprise. Voir chapitres 6 et 7.

La sonde envoie-t-elle elle-même des alertes par courriel ou messagerie ? Non. Les alertes sont produites par vos plateformes d'observabilité ou par Nagios et Centreon à partir des données de la sonde. Voir chapitre 18.

Combien de temps les résultats sont-ils conservés ? Par défaut, 14 jours pour les exécutions détaillées, 400 jours pour les agrégats horaires et 7 jours pour les artefacts. Ces valeurs sont configurables. Voir chapitre 9.

Que faire d'une sonde en fin de vie ? Révoquez les identifiants qu'elle utilisait, réinitialisez-la et détruisez ou effacez de manière sécurisée ses supports de stockage (microSD et, le cas échéant, SSD). Voir chapitre 20, § 20.13.

21.3 Annexe C — Récapitulatif des prérequis

Le détail figure au chapitre 4.

> **Information** — Cette annexe est un aide-mémoire des prérequis et de la configuration matérielle minimale de chaque édition. Elle ne dimensionne pas une sonde : la méthode de calcul de la charge, les seuils et le nombre de contrôles qu'une sonde peut exécuter sont décrits au chapitre 12 (§ 12.6, « Charge et dimensionnement »).

21.3.1 Matériel

Élément	Essential	Professional	Enterprise
Carte	Raspberry Pi 5	Raspberry Pi 5	Raspberry Pi 5
Mémoire vive	2 Go minimum, 4 Go recommandés	4 Go minimum	8 Go minimum
Stockage	microSD préconfigurée 32 Go, High Endurance, fournie par l'éditeur	Idem	Idem
Stockage complémentaire (facultatif)	SSD NVMe M.2 2230 ou 2242 de la liste des modèles pris en charge, sur carte M.2 HAT+ officielle de Raspberry Pi	Idem	Idem ; surtout utile pour cette édition
Alimentation	Officielle 27 W USB-C (5,1 V, 5 A) recommandée ; obligatoire avec un SSD NVMe	Idem	Officielle 27 W fortement recommandée ; obligatoire avec un SSD NVMe
Refroidissement	Refroidisseur actif officiel ou boîtier ventilé recommandé	Idem	Refroidisseur actif officiel ou boîtier ventilé **obligatoire**
Réseau	Port Ethernet et câble RJ45	Idem	Idem
Boîtier	Recommandé	Recommandé	Recommandé, ventilé

Dans l'offre « boîtier complet », le Raspberry Pi 5, le boîtier, le refroidissement et la carte microSD sont fournis et assemblés. Les autres éléments (alimentation, câble Ethernet, accessoires, SSD) ne sont fournis que s'ils figurent sur la liste de colisage de votre commande (chapitre 4, § 4.1.3).

Avec un SSD, l'alimentation officielle de 27 W est requise. Le refroidisseur actif (*Active Cooler*) est compatible avec la carte M.2 HAT+ et s'installe en premier. Le boîtier officiel du Raspberry Pi 5 n'est alors utilisable que sans couvercle ni ventilateur. La liste des modèles de SSD pris en charge figure dans les notes de version. Voir chapitres 4 et 5.

21.3.2 Réseau

Élément	Exigence
Adressage	Serveur DHCP, ou adresse IP statique réservée avec masque, passerelle et serveurs DNS. IPv4 obligatoire
IPv6 (facultatif)	Désactivé par défaut. Si activé : autoconfiguration (SLAAC) sur le lien, ou adresse statique (notation CIDR) et passerelle ; ICMPv6 autorisé pour la découverte des voisins
Port de commutateur	Port d'accès dans le VLAN de supervision : la sonde n'étiquette pas ses trames
802.1X (si le port est contrôlé)	Serveur RADIUS acceptant EAP-TLS ; certificat client émis pour la sonde (demande générée sur la sonde, recommandé, ou fichier PKCS#12) ; certificat de l'autorité de certification du serveur RADIUS. À défaut, port non contrôlé ou exception par adresse MAC (MAB) pour la première mise en service
DNS	Serveurs, en IPv4 ou en IPv6, capables de résoudre les noms des ressources supervisées
NTP	Accès à au moins un serveur NTP (UDP 123)
Proxy	Adresse, port et éventuels identifiants si les flux sortants passent par un proxy
Pare-feu	Flux de l'annexe E ouverts
Poste d'administration	Navigateur récent, sur un réseau autorisé à joindre la sonde en HTTPS

21.3.3 Organisation et sécurité

Élément	Exigence
Édition	Édition indiquée sur l'encart ou l'étiquette, conforme à la commande
Comptes de test	Un compte dédié par application, sans données réelles, droits minimaux, sans authentification multifacteur dans l'environnement supervisé
Comptes de base de données	Lecture seule, limités aux objets interrogés
Clés d'ingestion	Pour chaque plateforme d'observabilité retenue, avec les droits d'ingestion requis
Certificat (facultatif)	Certificat émis par l'autorité de l'entreprise pour le nom de la sonde
Code de récupération	Coffre-fort de mots de passe de l'organisation prêt à recevoir le code de récupération, affiché une seule fois à la fin de la mise en service
Contrat de maintenance (facultatif)	Justificatif de maintenance fourni par l'éditeur, à importer sur la sonde pour accéder aux nouvelles versions fonctionnelles
Lieu d'installation	Local technique ou baie, ventilé, à l'abri de l'humidité, idéalement sur alimentation secourue

21.4 Annexe D — Checklist de mise en service

Cochez chaque point au fur et à mesure. Les renvois indiquent la procédure détaillée.

Avant l'installation

- [ ] Édition acquise conforme à la commande ; encart ou étiquette (code d'installation) disponible (chapitre 6)
- [ ] Matériel conforme à l'édition, ou boîtier complet reçu (chapitre 4, annexe C)
- [ ] Emplacement choisi : ventilation, alimentation secourue, port réseau disponible (chapitre 5)
- [ ] Port de commutateur configuré en mode accès dans le bon VLAN (chapitre 7)
- [ ] Si le port est soumis à 802.1X : certificat client (ou demande à générer sur la sonde) et certificat de l'AC du serveur RADIUS préparés, ou port non contrôlé / exception MAB prévus pour la première mise en service (chapitre 7)
- [ ] Décision prise sur l'activation d'IPv6 et, le cas échéant, paramètres définis : SLAAC ou adresse statique (chapitre 7)
- [ ] Adresse IP réservée en DHCP ou attribuée en statique (chapitre 7)
- [ ] Flux réseau ouverts conformément à l'annexe E
- [ ] Paramètres de proxy connus, le cas échéant (chapitre 7)
- [ ] Comptes de test et comptes de base de données créés (chapitres 9 et 11)
- [ ] Méthode de mise en service choisie : fichier de provisionnement ou assistant (chapitre 6)
- [ ] Le cas échéant, `provision.yaml` préparé, validé et copié à la racine de la partition de démarrage, avec les fichiers 802.1X qu'il désigne (chapitres 6 et 7)
- [ ] Coffre-fort de l'organisation prêt à recevoir le code de récupération (chapitre 6)

Installation physique

- [ ] Carte microSD insérée hors tension, contacts tournés vers la carte (chapitre 5)
- [ ] Refroidisseur et boîtier installés (chapitre 5)
- [ ] Le cas échéant, SSD NVMe monté sur la carte M.2 HAT+, après le refroidisseur actif ; boîtier officiel sans couvercle ni ventilateur (chapitre 5)
- [ ] Câble Ethernet branché avant la mise sous tension (chapitre 5)
- [ ] Alimentation conforme branchée (chapitre 5)
- [ ] Voyants conformes au comportement attendu (chapitre 5)

Mise en service

- [ ] Adresse de la sonde identifiée (chapitre 6)
- [ ] Empreinte du certificat vérifiée à la première connexion (chapitre 6)
- [ ] Assistant terminé ou provisionnement confirmé par `provision-result.txt` (chapitre 6)
- [ ] `provision.yaml` absent de la partition de démarrage après provisionnement, ainsi que les fichiers 802.1X (`pkcs12File`, `caFile`) (chapitre 6)
- [ ] Le cas échéant, authentification 802.1X réussie : état de santé `network` normal (chapitre 7)
- [ ] Édition correcte affichée (chapitre 6)
- [ ] Heure correcte et synchronisation NTP active (chapitre 7)
- [ ] Compte administrateur nominatif créé ; second administrateur prévu (chapitre 9)
- [ ] Certificat de l'entreprise installé, le cas échéant (chapitre 7)
- [ ] SSH laissé désactivé, sauf besoin justifié ; verrouillage définitif décidé selon votre politique de sécurité (chapitre 7)
- [ ] Code de récupération affiché à la fin de la mise en service, conservé dans le coffre-fort de l'organisation (chapitre 6)

- [ ] Le cas échéant, SSD activé et état de santé `storage` normal (chapitre 20)

Configuration de la supervision

- [ ] Secrets enregistrés (chapitre 9)
- [ ] Premier contrôle créé et exécution immédiate réussie (chapitre 6)
- [ ] Contrôles techniques configurés (chapitre 10)
- [ ] Contrôles applicatifs et métier configurés (chapitre 11)
- [ ] Scénarios navigateur testés en exécution d'essai (chapitres 14 et 15)
- [ ] Plans d'exécution et fenêtres d'activité définis (chapitre 12)
- [ ] Seuils, timeouts et nouvelles tentatives ajustés (chapitre 8)

Intégration et exploitation

- [ ] Destinations d'export créées et testées (chapitre 18)
- [ ] Données visibles dans la ou les plateformes (chapitre 18)
- [ ] Tableaux de bord et alertes créés ou importés, par exemple à partir des modèles Grafana et Prometheus fournis (chapitre 18)
- [ ] Alerte « sonde muette » configurée (chapitre 20)
- [ ] Services Nagios ou Centreon `check_sva` configurés, le cas échéant (chapitre 18)
- [ ] Politique de mises à jour définie : mode de chaque canal (correctifs de sécurité, versions fonctionnelles) et fenêtres de maintenance compatibles avec vos plans d'exécution, ou mode manuel assumé ; état de santé `updates` vérifié (chapitre 20)
- [ ] Alertes neutralisées pendant les fenêtres de maintenance sur vos plateformes (Alertmanager, Datadog, Dynatrace, Centreon, Nagios) ; pour un environnement critique, plusieurs sondes par site avec des fenêtres sans chevauchement (chapitres 18 et 20)
- [ ] Date de fin de la période de support relevée (`supportEndsAt`) ; justificatif de maintenance importé si un contrat a été souscrit (chapitre 20)
- [ ] Premier export de configuration réalisé et archivé hors de la sonde (chapitre 20)
- [ ] Sonde inscrite à l'inventaire : emplacement, adresse, numéro de série, édition, responsable
- [ ] Contacts du support connus des équipes d'exploitation (annexe L)

21.5 Annexe E — Tableau des flux réseau

Ce tableau reprend, à l'identique, la matrice des flux de la configuration réseau, pour la transmettre à l'équipe chargée des pare-feu.

Sens	Source	Destination	Port / protocole	Usage	Obligatoire
Entrant	Postes d'administration	Sonde	443/TCP (HTTPS)	Interface d'administration, API	Oui
Entrant	Serveur Prometheus	Sonde	443/TCP (HTTPS)	Collecte de `/metrics`	Si Prometheus en mode collecte
Entrant	Serveur Nagios ou Centreon	Sonde	443/TCP (HTTPS)	Plugin `check_sva`	Si Nagios ou Centreon
Entrant	Postes d'administration	Sonde	80/TCP (HTTP)	Redirection permanente vers HTTPS, aucun contenu servi	Non (désactivable)
Entrant	Postes du réseau local	Sonde	5353/UDP (mDNS)	Résolution du nom `.local` de la sonde ; annonce active par défaut	Non (désactivable dans la section `network`)

Sens	Source	Destination	Port / protocole	Usage	Obligatoire
Entrant	Postes d'administration autorisés	Sonde	22/TCP (SSH)	Maintenance avancée	Non : désactivé par défaut ; activation temporaire, 4 heures par défaut, 72 heures au maximum (§ 7.9.4)
Couche 2 (sans flux IP)	Sonde	Commutateur (authentificateur 802.1X)	EAPOL (*EAP over LAN*) : trames Ethernet de type 0x888E, aucun port ni adresse IP	Authentification 802.1X EAP-TLS du port ; le commutateur relaie vers le serveur RADIUS	Si le port est contrôlé par 802.1X (§ 7.2.6)
Entrant et sortant	Sonde	Routeurs et voisins du lien local	ICMPv6 (découverte des voisins, NDP, dont les annonces de routeur utilisées en mode `slaac`)	Adressage et résolution d'adresses IPv6	Si l'IPv6 est activé (§ 7.2.5)
Sortant	Sonde	Serveurs DHCP	67-68/UDP	Adressage	Si DHCP
Sortant	Sonde	Serveurs DNS	53/UDP et 53/TCP	Résolution de noms	Oui
Sortant	Sonde	Serveurs NTP	123/UDP	Synchronisation de l'heure	Oui
Sortant	Sonde	Dépôt de mises à jour de l'éditeur, ou miroir interne	443/TCP (HTTPS)	Mises à jour en ligne : vérification et téléchargement des images signées ; adresse par défaut `https://updates-synovex.incitypark.fr`, inscrite dans le profil éditeur	Si mises à jour en ligne
Sortant	Sonde	Proxy de l'entreprise	Port du proxy (par exemple 3128/TCP)	Relais des flux HTTP(S) sortants (§ 7.5)	Si proxy
Sortant	Sonde	Plateformes SaaS (Datadog, Dynatrace, Grafana Cloud)	443/TCP (HTTPS, OTLP/HTTP)	Export OpenTelemetry	Si configurées
Sortant	Sonde	Collecteur OpenTelemetry ou Agent Datadog du client	4318/TCP (OTLP/HTTP) ou 4317/TCP (OTLP/gRPC), selon le protocole de la destination (chapitre 18, § 18.1.4)	Export OpenTelemetry	Si configurés
Sortant	Sonde	Proxy inverse placé devant le récepteur OTLP de Prometheus	Port du proxy inverse (par exemple 443/TCP, HTTPS)	Export des métriques en mode envoi (chapitre 18, § 18.5.3)	Si configuré
Sortant	Sonde	Ressources supervisées	Selon les contrôles : ICMP ou ICMPv6 ; ports TCP des services ; 53 ; 80 et 443 ; SGBD : 5432 (PostgreSQL), 3306 (MySQL/MariaDB), 1433 (SQL Server), 1521 (Oracle)	Contrôles	Oui, selon les contrôles
Sortant	Sonde	Console centrale	443/TCP (HTTPS, authentification mutuelle mTLS)	Gestion multi-sondes	Non : évolution prévue, facultative



Information — Les ports des SGBD indiqués sont les ports par défaut. Ils peuvent différer dans votre environnement. Les ports des contrôles sont ceux que vous configurez. La sonde n'utilise aucun serveur de licences en ligne et n'envoie aucune télémétrie à l'éditeur.

> **Information** — Si l'IPv6 est activé, les flux de ce tableau s'appliquent à l'identique en IPv6 : le pare-feu local de la sonde applique les mêmes règles dans les deux familles d'adresses. Les échanges RADIUS ont lieu entre le commutateur et le serveur RADIUS : ils ne partent pas de la sonde. Le mécanisme de mise à jour A/B ne crée aucun flux supplémentaire, et une mise à jour hors ligne n'en exige aucun vers l'éditeur.

21.6 Annexe F — Référence de l'API REST

Base : `https://<sonde>/api/v1`. Authentification : `Authorization: Bearer <jeton>` ou session de l'interface. Dans les exemples de la documentation, le jeton n'apparaît jamais dans la ligne de commande : l'en-tête est lu dans un fichier protégé, avec `curl -H @"$HOME/.sva/entete-auth"` (chapitre 9, § 9.3.3). L'authentification par certificat client (mTLS) est reportée à une version ultérieure, avec la console centrale. Description complète au format OpenAPI 3.1 : `GET /api/v1/openapi.json`. Voir chapitre 9.

21.6.1 Conventions

Sujet	Règle
Format	JSON UTF-8 ; dates RFC 3339 en UTC ; durées en millisecondes dans les champs suffixés `Ms` (`durationMs`, `p50DurationMs`…) des réponses
Erreurs	`application/problem+json` (RFC 9457) : `type` (adresse relative `/api/v1/problems/{code}`), `title`, `status`, `detail`, `errors[]`. `400` pour un JSON mal formé ; `422` pour une erreur de validation ; `403` pour un rôle insuffisant, avec le type `/api/v1/problems/edition-feature-unavailable` pour une fonction absente de l'édition, ou avec le type `/api/v1/problems/current-password-invalid` pour un mot de passe actuel erroné (`PUT /api/v1/users/me/password`) ; `405` pour une méthode non admise (section `license` en lecture seule)
Concurrence	`ETag` sur les ressources de configuration, renvoyé par leur lecture (pour un secret ou un compte : `GET /api/v1/secrets/{name}` ou `GET /api/v1/users/{id}`) ; `If-Match` obligatoire en `PUT`, `PATCH`, `DELETE` sur une ressource existante ; `If-None-Match: *` pour une création par `PUT` (secret) ; réponse `428` si la requête ne présente ni `If-Match` ni `If-None-Match`, `412` en cas de conflit de version ou de création d'une ressource qui existe déjà. Les requêtes d'action et d'import en écriture seule (`PUT /api/v1/system/maintenance`, `/system/certificate`, `/system/dot1x/credential`) ne portent pas sur une ressource de configuration et n'en relèvent pas. Exception : `PUT /api/v1/users/me/password` n'exige ni `If-Match` ni `If-None-Match`, la preuve étant le mot de passe actuel (chapitre 9, § 9.3.5)
Pagination	`?limit=…&cursor=…` ; 100 éléments par défaut, 1 000 au maximum ; curseur opaque en base64url ; réponse `{ "items": [...], "nextCursor": "…" }`. Ordre total (`startedAt`, `id`) ; curseur valide 24 heures (au-delà : `410 Gone`) ; pour un export reproductible, fixez une borne `to` ; les éléments purgés par la rétention pendant la lecture disparaissent des pages suivantes, sans instantané (chapitre 17)
Actions longues	Réponse `202 Accepted` et en-tête `Location` vers la ressource de suivi
Paramètres de chemin	En anglais : `{id}` pour les contrôles, plans, utilisateurs, jetons et destinations d'export ; `{name}` pour les secrets, scripts, plugins et artefacts ; `{runId}` pour les exécutions ; `{section}` pour les sections de paramètres

21.6.2 Ressources

Méthode et chemin	Rôle minimal	Description
`GET /health`	aucun	Vivacité du moteur principal : `OK` (code `200`) ; `MAINTENANCE` (code `503`) pendant l'état de maintenance qui ouvre l'activation d'une mise à jour
`GET /metrics`	`metrics`	Exposition des métriques au format Prometheus

Méthode et chemin	Rôle minimal	Description
`GET /api/v1/system`	`viewer`	Version, édition et état du certificat d'édition (`editionCertificateStatus` : mode restreint et sa cause), ressources (dont `resources.memoryTotalBytes`), état des modules et des exports, états de santé (dont `storage`, `network`, `support`, `updates`), date de fin de la période de support (`supportEndsAt`) et du justificatif de maintenance (`maintenanceEndsAt`). `supportEndsAt` est lisible avec le rôle `viewer`
`GET /api/v1/checks`	`viewer`	Liste des contrôles (filtres `type`, `enabled`, `tag`, `status`)
`POST /api/v1/checks`	`operator` ; `admin` pour un scénario scripté ou un plugin personnalisé	Création d'un contrôle ; un `operator` qui crée un scénario scripté ou un contrôle utilisant un plugin personnalisé reçoit `403`
`GET /api/v1/checks/{id}`	`viewer`	Détail d'un contrôle
`PUT /api/v1/checks/{id}`	`operator` pour un contrôle non scripté ; `admin` pour un scénario scripté ou un contrôle utilisant un plugin personnalisé	Remplacement complet ; un `operator` qui crée, modifie ou retire `browser.mode: script`, `browser.scriptRef` ou un plugin personnalisé reçoit `403` (chapitre 9, § 9.6.1)
`PATCH /api/v1/checks/{id}`	`operator` ; `admin` pour toute modification de `browser.mode: script`, de `browser.scriptRef` ou d'un plugin personnalisé	Modification partielle (JSON Merge Patch) ; un `operator` qui crée, modifie ou retire `browser.mode: script`, `browser.scriptRef` ou un plugin personnalisé reçoit `403`
`DELETE /api/v1/checks/{id}`	`operator`	Suppression (historique conservé selon la rétention) ; refusée avec `409` et la liste des plans concernés si un plan référence le contrôle
`POST /api/v1/checks/{id}/enable`	`operator`	Activation
`POST /api/v1/checks/{id}/disable`	`operator`	Désactivation
`POST /api/v1/checks/{id}/runs`	`operator`	Exécution immédiate ; `?dryRun=true` pour une exécution d'essai
`GET /api/v1/checks/{id}/status`	`metrics`	Dernier état consolidé
`GET /api/v1/checks/{id}/aggregates?from=&to=&step=1h`	`viewer`	Agrégats horaires d'un contrôle
`GET /api/v1/runs`	`viewer`	Historique (filtres `checkId`, `planRunId`, `status`, `from`, `to`)
`GET /api/v1/runs/{runId}`	`viewer`	Détail d'une exécution
`GET /api/v1/runs/{runId}/artifacts/{name}`	`viewer`	Téléchargement d'un artefact
`GET /api/v1/plans`, `GET /api/v1/plans/{id}`	`viewer`	Plans d'exécution

Méthode et chemin	Rôle minimal	Description
`POST`, `PUT`, `PATCH`, `DELETE /api/v1/plans[/{id}]`	`operator`	Création, modification, suppression d'un plan
`POST /api/v1/plans/{id}/enable` · `/disable`	`operator`	Activation, désactivation d'un plan
`POST /api/v1/plans/{id}/runs`	`operator`	Exécution immédiate d'un plan
`POST /api/v1/scripts`	`admin`	Téléversement d'un script (renvoie une référence `nom@sha256:…`)
`GET /api/v1/scripts[/{name}]`	`viewer`	Liste des scripts et de leurs versions
`GET /api/v1/secrets`	`admin`	Noms et métadonnées des secrets (jamais les valeurs)
`GET /api/v1/secrets/{name}`	`admin`	Métadonnées d'un secret, dont `allowedHosts`, et en-tête `ETag`, à renvoyer dans `If-Match` lors d'une rotation ou d'une suppression ; la valeur du secret n'est jamais renvoyée
`PUT /api/v1/secrets/{name}`	`admin`	Création ou rotation d'un secret (écriture seule), corps `{"value": …, "allowedHosts": […]}` : la liste des hôtes autorisés est obligatoire, et la sonde n'injecte le secret que vers ces hôtes (chapitre 9, § 9.8). Création : en-tête `If-None-Match: *` (`412` si le secret existe déjà) ; rotation : en-tête `If-Match` avec l'`ETag` du secret. Lors d'une rotation, renvoyez la liste `allowedHosts` relue, avec l'`ETag`, par `GET /api/v1/secrets/{name}` (§ 9.8.4)
`DELETE /api/v1/secrets/{name}`	`admin`	Suppression (refusée avec `409` si un contrôle le référence)
`GET`, `POST`, `PUT`, `DELETE /api/v1/exporters[/{id}]`	`admin`	Destinations d'export
`POST /api/v1/exporters/{id}/test`	`admin`	Test d'une destination
`GET`, `PUT /api/v1/settings/{section}`	`admin`	Paramètres : `network`, `time`, `proxy`, `retention`, `security`, `trust`, `logging`, `updates`, `ssh`, `dot1x`, `license` (lecture seule : `PUT` refusé avec `405`). Le certificat de la sonde se gère par `POST /api/v1/system/certificate/csr` (demande générée sur la sonde) et `PUT /api/v1/system/certificate` (import)
`POST /api/v1/settings/network/apply`	`admin`	Application d'une configuration réseau ; renvoie un identifiant de confirmation
`POST /api/v1/settings/network/confirm`	`admin`	Confirmation, depuis la nouvelle adresse, sous 120 secondes ; sans confirmation, l'ancienne configuration est rétablie
`POST /api/v1/settings/ssh/lock`	`admin`	Verrouillage définitif de SSH (corps `{"confirm": "<nom d'hôte>"}`) ; irréversible jusqu'à une réinitialisation usine
`GET`, `POST /api/v1/users`	`admin`	Liste et création des comptes de l'interface (chapitre 9, § 9.4.2)

Méthode et chemin	Rôle minimal	Description
`GET /api/v1/users/{id}`	`admin`	Lecture d'un compte (identifiant, identifiant de connexion, rôle ; jamais le mot de passe ni son empreinte) et en-tête `ETag`, à renvoyer dans `If-Match` pour `PATCH /api/v1/users/{id}`
`PATCH /api/v1/users/{id}`	`admin`	Modification du rôle (`role`) ou du mot de passe (`password`) d'un compte, avec `If-Match` ; l'identifiant de connexion ne se modifie pas. Un changement de rôle ferme immédiatement les sessions de l'interface du compte ; un changement de mot de passe les ferme également, sauf la session depuis laquelle la requête est émise si elle appartient au compte. Les jetons d'API du compte ne sont pas modifiés et conservent le rôle fixé à leur création (chapitre 9, § 9.4.2)
`DELETE /api/v1/users/{id}`	`admin`	Suppression d'un compte ; ses sessions et ses jetons sont révoqués immédiatement
`PUT /api/v1/users/me/password`	tout utilisateur authentifié	Changement de son propre mot de passe : corps `{"currentPassword": …, "newPassword": …}`. Sans `If-Match` (exception à la règle de concurrence). Mot de passe actuel erroné : `403` avec le type `/api/v1/problems/current-password-invalid`. Après le changement, les autres sessions de l'interface du compte sont fermées, la session courante est conservée, et les jetons d'API du compte ne sont pas modifiés
`GET`, `POST /api/v1/tokens`	`admin`	Liste et création des jetons d'API ; un jeton est immuable (ni `PUT` ni `PATCH`) : pour le modifier, créez-en un nouveau et révoquez l'ancien
`DELETE /api/v1/tokens/{id}`	`admin`	Révocation d'un jeton
`GET /api/v1/audit-events`	`admin`	Journal d'audit (filtres `from`, `to`, `actor`, `action`)
`GET /api/v1/config/export`	`admin`	Export de la configuration (archive JSON, sans section `network`, sans comptes ni secrets)
`POST /api/v1/config/export`	`admin`	Export avec la phrase de passe dans l'en-tête `X-Config-Passphrase`, comme pour l'import : comptes (empreintes) et secrets inclus, chiffrés en AES-256-GCM
`POST /api/v1/config/import`	`admin`	Import de la configuration ; `?dryRun=true` pour valider ; identité de la sonde restaurée par défaut, paramètre `?newIdentity=true` pour cloner ; phrase de passe dans l'en-tête `X-Config-Passphrase` ; compte rendu JSON
`GET /api/v1/events`	`viewer`	Flux d'événements en temps réel (Server-Sent Events)

Méthode et chemin	Rôle minimal	Description
`GET /api/v1/updates`	`admin`	État des mises à jour : version de chaque emplacement système, emplacement actif, mises à jour disponibles, `supportEndsAt`, `maintenanceEndsAt` ; état de la mise à jour en attente (`state`), activation annoncée (`scheduledActivation` : identifiant `activationId` , numéro d'annonce `sequence` , cas `kind` , version, canal, caractère critique, heure prévue, fin prévue `windowEnd` , `noticeReduced`), report en cours (`postponedUntil`) ; image préparée dans l'emplacement inactif (`staged` : version, canal, caractère critique) ; dernière tentative (`lastResult` : identifiant d'activation, version, résultat, dates)
`POST /api/v1/updates/apply`	`admin`	Activation de la mise à jour installée sur l'emplacement inactif : immédiate, ou programmée avec le corps facultatif `{"scheduledAt": "<date RFC 3339>"}` . Drainage, redémarrage, retour automatique à l'emplacement précédent si la sonde n'est pas saine dans les 10 minutes. Sans `scheduledAt` , activation immédiate et sans préavis ; date passée ou à moins d'une minute : `422` ; date antérieure à l'expiration du préavis : acceptée, réponse avec `noticeReduced: true` . La réponse porte l'identifiant `activationId` , à comparer ensuite avec `lastResult.activationId` (chapitre 20, § 20.4.2 et § 20.4.12)
`POST /api/v1/updates/postpone`	`admin`	Report de l'activation prévue, corps `{"until": "<date RFC 3339>"}` : 30 jours au plus, 7 jours pour un correctif critique ; journalisé
`POST /api/v1/updates/upload`	`admin`	Import d'une image de mise à jour signée (mise à jour hors ligne)
`POST /api/v1/system/reboot`	`admin`	Redémarrage propre
`POST /api/v1/system/shutdown`	`admin`	Arrêt propre
`POST /api/v1/system/factory-reset`	`admin`	Réinitialisation usine (corps `{"confirm": "<nom d'hôte>"}`)
`PUT /api/v1/system/maintenance`	`admin`	Import du justificatif de maintenance (fichier signé)
`POST /api/v1/system/recovery-code`	`admin`	Génère un nouveau code de récupération (mot de passe de l'administrateur exigé) ; affiché une seule fois, l'ancien est invalidé (chapitre 6, § 6.4.6)
`GET /api/v1/plugins`	`operator`	Liste des plugins personnalisés déposés (Professional, Enterprise)
`POST /api/v1/plugins?name=<nom>` , `DELETE /api/v1/plugins/{name}`	`admin`	Dépôt d'un plugin personnalisé (fichier exécutable, 1 Mo au maximum) ; retrait refusé (`409`) s'il est utilisé par un contrôle (chapitre 10, § 10.7.10)
`POST /api/v1/system/storage/ssd`	`admin`	Activation du SSD NVMe (corps `{"confirm": "<nom d'hôte>"}`) : formatage, migration des données, redémarrage
`POST /api/v1/system/dot1x/csr`	`admin`	Génération sur la sonde de la clé et de la demande de certificat client 802.1X (méthode recommandée)

Méthode et chemin	Rôle minimal	Description
`PUT /api/v1/system/dot1x/credential`	`admin`	Installation du certificat client signé (PEM) émis pour la dernière CSR, ou import d'un PKCS#12 protégé ; écriture seule
`POST /api/v1/system/diagnostics`	`admin`	Archive de diagnostic : version, santé, configuration sans secrets, journaux récents
`GET /api/v1/system/certificate`	aucun	Certificat TLS de la sonde (PEM) ; vérifier son empreinte avant usage
`GET /api/v1/openapi.json`	tout utilisateur authentifié	Description OpenAPI

21.6.3 Rôles

Rôle	Droits
`admin`	Tous les droits : comptes, jetons, secrets, réseau (dont 802.1X), SSH, mises à jour (politique, activation, report), justificatif de maintenance, stockage SSD, scénarios scriptés, plugins personnalisés
`operator`	Création et modification des contrôles déclaratifs et des plans, activation, désactivation, déclenchement, consultation. Pas de code exécutable ; référence d'un secret existant possible, sans le lire, le créer, le modifier ni changer ses hôtes autorisés : toute requête qui crée, modifie ou retire `browser.mode: script`, `browser.scriptRef` ou un plugin personnalisé est refusée avec `403`
`viewer`	Lecture seule des états, résultats et artefacts
`metrics`	`/metrics` et `GET /api/v1/checks/{id}/status` uniquement

21.7 Annexe G — Référence des configurations

Les tableaux ci-dessous résument les champs. Les schémas complets et les exemples figurent dans les chapitres indiqués.

21.7.1 Champs communs à tous les contrôles (chapitre 8)

Champ	Type	Obligatoire	Description
`id`	texte	Oui	Identifiant stable : minuscules, chiffres, tirets, 2 à 63 caractères
`name`	texte	Oui	Libellé lisible (120 caractères maximum)
`description`	texte	Non	Description libre
`type`	énumération	Oui	`icmp`, `tcp`, `dns`, `http`, `api`, `nagios`, `database`, `browser`
`enabled`	booléen	Non (`true`)	Activation
`tags`	objet	Non	8 étiquettes maximum
`schedule`	objet ou `null`	Non	`interval` (`5m`) **ou** `cron` + `timezone`, `jitter` ; `null` : exécution par un plan ou manuelle
`timeout`	durée	Non	Délai maximal de l'exécution
`retry`	objet	Non	`attempts` (0 à 5), `interval`
`thresholds`	objet	Non	`warningDuration`, `criticalDuration`

Champ	Type	Obligatoire	Description
`secrets`	liste	Non	Noms des secrets autorisés (16 maximum)
`ipFamily`	énumération	Non (`auto`)	Famille d'adresses utilisée pour joindre la cible : `auto` (IPv6 si l'IPv6 est activé sur la sonde et si la cible a une adresse AAAA, sinon IPv4), `ipv4` ou `ipv6`

Durées : `500ms`, `30s`, `5m`, `1h`. Références : `${secret:NOM}`, `${param:nom}`, `${var:nom}`.

21.7.2 Champs propres à chaque type

Type	Champs principaux	Chapitre
`icmp`	`host`, `count` (1 à 20), `maxPacketLossPercent`, `warningRttMs`, `criticalRttMs`	10
`tcp`	`host`, `port`, `tls`, `send`, `expect`	10
`dns`	`name`, `recordType` (`A`, `AAAA`, `CNAME`, `MX`, `TXT`, `NS`, `SRV`, `PTR`, `SOA`), `servers`, `expect`	10
`http`	`method`, `url`, `headers`, `body`, `followRedirects`, `tls` (`verify`, `caRef`, `certificateExpiryWarningDays`, `certificateExpiryCriticalDays`), `proxy` (`"none"` pour contourner le proxy global), `assertions` (`kind` : `status`, `header`, `body`, `jsonPath`, `xpath`, `duration` ; `path`, `operator`, `value`)	10
`api`	`requests` (1 à 20) : champs de `http` + `name`, `extract` (`type` : `jsonpath`, `xpath`, `regex` ; `reveal` pour afficher la valeur, masquée par défaut)	11
`nagios`	`plugin` (nom sans chemin), `args`, `perfdataExport`	10
`database`	`engine` (`mysql`, `mariadb`, `postgresql`, `sqlserver`, `oracle`), `host`, `port`, `database`, `username`, `passwordSecret`, `query` **ou** `procedure`, `params`, `rules` (`column`, `operator` : `eq`, `ne`, `lt`, `lte`, `gt`, `gte` ; `value`, `severity`, `scope`), `onEmpty` (`UNKNOWN` par défaut), `exportValues` (première ligne), `tls`, `properties` (liste fermée de propriétés JDBC)	11
`browser`	`mode` (`steps` ou `script`), `steps` (50 maximum), `scriptRef`, `params`, `viewport`, `userAgentSuffix`, `captureOnFailure`, `blockDomains`	14

Opérateurs d'assertion : `eq`, `ne`, `lt`, `lte`, `gt`, `gte`, `contains`, `notContains`, `matches`, `exists`.

Actions des scénarios déclaratifs : `goto`, `click`, `fill`, `select`, `press`, `hover`, `waitForSelector`, `waitForNavigation`, `assertVisible`, `assertHidden`, `assertText`, `assertUrl`, `assertTitle`. Propriétés d'une étape : `name`, `action`, `url`, `selector`, `frame`, `value`, `key`, `operator`, `timeout`, `waitForNavigation`.

21.7.3 Plan d'exécution (chapitre 12)

Champ	Description
`id`, `name`, `enabled`	Identification et activation
`schedule`	`interval` ou `cron` (obligatoire)
`activeWindows`	Liste de plages : `days` (`mon` à `sun`), `from`, `to` (`HH:MM`)
`stopOnFailure`	Arrêt du plan au premier échec
`maxDuration`	Durée maximale du plan
`items`	Liste ordonnée (1 à 50) : `checkId`, `continueOnFailure`

21.7.4 Destination d'export (chapitre 18)

Champ	Description
`id`, `kind` (`otlp` ou `prometheus`), `enabled`, `preset`	Identification ; préréglage : `otlp-generic`, `datadog-agent`, `datadog-agentless`, `dynatrace`, `grafana-cloud`, `prometheus-otlp`
`signals`	`metrics`, `traces`, `logs`
`endpoint` ou `signalEndpoints`	URL de base, ou URL complète par signal
`protocol`	`http/protobuf` (par défaut), `http/json`, `grpc`
`headers`	En-têtes ; les valeurs sensibles référencent un secret
`compression`, `timeout`, `tls`	`gzip` ou `none` ; délai ; `caRef`, `clientCertRef`, `insecureSkipVerify`
`metrics`	`temporality` (`cumulative` ou `delta`), `exportInterval`, `histogram`, `stepHistograms`
`logs`	`audit` : export du journal d'audit (`false` par défaut)
`retry`, `queue`	`maxElapsed` ; `maxBatches`
`path`, `requireToken`	Pour `kind: prometheus`

21.7.5 Fichier de provisionnement `provision.yaml` (chapitre 6)

Clé	Description
`version`	Version du format (`1`)
`hostname`	Nom d'hôte de la sonde
`site`	Nom du site (attribut `sva.site`)
`admin.username`, `admin.passwordHash`	Compte administrateur ; empreinte du mot de passe, jamais le mot de passe en clair. Format : Argon2id au format PHC (`$argon2id$v=19$m=…`)
`network.mode`	`dhcp` ou `static`
`network.address`, `network.gateway`, `network.dns`, `network.ntp`	Paramètres réseau ; les serveurs DNS peuvent être IPv4 ou IPv6
`network.ipv6.enabled`, `network.ipv6.mode`, `network.ipv6.address`, `network.ipv6.gateway`	IPv6 : activation (`false` par défaut), mode `slaac` ou `static`, adresse en notation CIDR et passerelle (mode `static`)
`dot1x.identity`, `dot1x.pkcs12File`, `dot1x.caFile`, `dot1x.pkcs12Password`	802.1X EAP-TLS avant tout accès au réseau : identité, fichier PKCS#12 de l'identifiant client et certificat de l'AC du serveur RADIUS, déposés sur la partition de démarrage, et mot de passe du fichier PKCS#12. Seul secret admis dans `provision.yaml` : fichiers et mot de passe sont supprimés après traitement
`ssh.locked`	`true` : verrouillage définitif de SSH, irréversible jusqu'à une réinitialisation usine
`updates.security.mode`, `updates.security.criticalPolicy`, `updates.security.criticalStagger`, `updates.feature.mode`, `updates.windows`, `updates.notice`, `updates.drainTimeout`	Politique de mises à jour dès le premier démarrage : mêmes champs et mêmes valeurs par défaut que la section `updates` (§ 21.7.7), **aucune fenêtre par défaut**
`proxy.url`, `proxy.noProxy`	Proxy sortant et exclusions

21.7.6 Valeurs par défaut

Paramètre	Valeur par défaut
Concurrence des contrôles légers	16, à titre indicatif
Concurrence plugins et bases de données	4, à titre indicatif
Concurrence navigateur	1 (2 configurable avec 8 Go), à titre indicatif
Intervalle entre nouvelles tentatives (`retry.interval`)	15 secondes
Durée maximale d'un plan d'exécution (`maxDuration`)	30 minutes
Export : `timeout` , `exportInterval`	10 secondes ; 60 secondes
Export : file d'attente (`queue.maxBatches`), taille d'un lot	100 lots ; 1 000 points par lot
Histogrammes par étape (`metrics.stepHistograms`)	Activés (`true`)
Scénarios : `viewport` , délai d'étape, délai de navigation	1366 × 768 ; 10 secondes ; 30 secondes
Contrôle `database` sans résultat (`onEmpty`)	`UNKNOWN` (`no_data`)
Pagination de l'API	100 éléments (1 000 au maximum)
Rétention des exécutions détaillées	14 jours
Rétention des agrégats horaires	400 jours
Rétention des artefacts	7 jours et 500 Mo
Rétention du journal d'audit	365 jours
Sauvegarde automatique de la configuration	Quotidienne, 3 générations
Délai de confirmation d'une configuration réseau	120 secondes
Annonce mDNS	Active (désactivable dans la section `network`)
Pare-feu local	Actif : entrées 443/TCP, 80/TCP (redirection), 5353/UDP (mDNS), 22/TCP si SSH est activé ; mêmes règles en IPv6
Port 80	Redirection permanente vers HTTPS (désactivable)
Sessions de l'interface	Expiration après 30 minutes d'inactivité ; durée absolue de 12 heures
Blocage après échecs de connexion	15 minutes après 5 échecs consécutifs, par compte et par adresse source
Saisie du code d'installation	5 essais, puis blocage de 15 minutes, doublé à chaque nouvelle série
Mots de passe des comptes	12 caractères minimum, 128 maximum
Journaux système	En mémoire (volatils) ; persistance temporaire (`logging.persistentUntil`) de 72 heures au plus
Famille d'adresses des contrôles (`ipFamily`)	`auto`
IPv6 (`network.ipv6.enabled`)	Désactivé (`false`)
802.1X (`dot1x.enabled`)	Désactivé ; avertissement 30 jours avant l'expiration du certificat client
Durée d'activation de SSH (`ssh.enabledUntil`)	4 heures (72 heures au maximum)
Page de récupération du compte administrateur	Ouverte 30 minutes après le démarrage ; 3 essais au plus par ouverture

Paramètre	Valeur par défaut
Code de réinitialisation usine (`factory-reset-challenge.txt`)	8 caractères, valable 24 heures
Activation des correctifs de sécurité (`updates.security.mode`)	`automatic`
Correctifs critiques (`updates.security.criticalPolicy`)	`next-window` (prochaine fenêtre)
Décalage des activations critiques (`updates.security.criticalStagger`)	`0` (aucun décalage)
Activation des versions fonctionnelles (`updates.feature.mode`)	`manual`
Fenêtres de maintenance (`updates.windows`)	Aucune : sans fenêtre, aucune activation automatique
Préavis avant activation (`updates.notice`)	24 heures (de 0 à 7 jours), pour une activation automatique ordinaire
Préavis d'un correctif critique	15 minutes (`next-window` et `asap`), augmentées du décalage `criticalStagger` en mode `asap`
Délai de drainage (`updates.drainTimeout`)	5 minutes (30 minutes au maximum)
Report d'une activation (`POST /api/v1/updates/postpone`)	30 jours au plus ; 7 jours au plus pour un correctif critique
Autotest après mise à jour	10 minutes accordées à l'autotest ; à défaut, retour automatique à l'emplacement précédent, après un redémarrage effectif
Période de support (correctifs de sécurité)	Au moins 5 ans après la mise sur le marché de la version majeure

21.7.7 Sections de paramètres système : réseau IPv6, 802.1X, SSH et mises à jour (chapitres 7 et 20)

Ces sections se consultent et se modifient par `GET` et `PUT /api/v1/settings/{section}` (rôle `admin`). Le tableau ne reprend que les champs introduits par les fonctions correspondantes.

Section et champ	Description
`network.ipv6.enabled`	Activation de l'IPv6 (`false` par défaut) ; IPv4 reste toujours actif
`network.ipv6.mode`	`slaac` (autoconfiguration) ou `static`
`network.ipv6.address` , `network.ipv6.gateway`	Adresse en notation CIDR et passerelle, en mode `static` . Le certificat autosigné de la sonde inclut l'adresse IPv6 statique
`dot1x.enabled`	Activation de l'authentification 802.1X EAP-TLS sur la prise Ethernet
`dot1x.identity`	Identité présentée au serveur RADIUS
`dot1x.caRef`	Référence, dans la section `trust` , du certificat de l'AC du serveur RADIUS
`ssh.enabled`	Activation de SSH (seul un `admin` peut l'activer)
`ssh.enabledUntil`	Échéance de l'activation, obligatoire : 4 heures par défaut, 72 heures au maximum. À l'échéance, SSH se désactive et les sessions sont fermées
`ssh.authorizedKeys`	Clés publiques autorisées
`ssh.locked`	Verrouillage définitif, en lecture seule ici : il se pose par `provision.yaml` ou par `POST /api/v1/settings/ssh/lock`

Section et champ	Description
`updates.security.mode`	Correctifs de sécurité : `automatic` (par défaut), activés dans une fenêtre de maintenance, ou `manual`
`updates.security.criticalPolicy`	Correctif d'une vulnérabilité activement exploitée : `next-window` (par défaut : au début de la première fenêtre qui commence au moins 15 minutes après l'annonce) ou `asap` (choix explicite : 15 minutes après l'annonce, augmentées de `criticalStagger`, même hors fenêtre) ; drainage dans les deux cas (chapitre 20, § 20.4.2)
`updates.security.criticalStagger`	Décalage fixe, propre à la sonde, ajouté au préavis de 15 minutes d'une activation `asap`, de `0` (par défaut) à `4h` : échelonne les sondes d'un même site, sans vérifier leur état (chapitre 20, § 20.4.11)
`updates.feature.mode`	Versions fonctionnelles, canal distinct : `manual` (par défaut) ou `automatic`
`updates.windows`	0 à 7 fenêtres de maintenance : `days` (`mon` à `sun`), `start` (`HH:MM`), `duration`. **Aucune fenêtre par défaut** : sans fenêtre, la mise à jour reste en attente et l'état de santé `updates` passe en `WARNING`
`updates.notice`	Préavis avant toute activation automatique : `24h` par défaut, de `0` à `7d` (activation manuelle et correctif critique : chapitre 20, § 20.4.2 et § 20.4.11)
`updates.drainTimeout`	Délai de drainage : `5m` par défaut, `30m` au maximum
`updates.repositoryUrl`	Facultatif : adresse d'un miroir interne du dépôt de mises à jour (HTTPS) ; les images restent signées et vérifiées

L'identifiant client 802.1X (clé privée et certificat) ne figure dans aucune section : il se génère par `POST /api/v1/system/dot1x/csr` ou s'importe par `PUT /api/v1/system/dot1x/credential`, en écriture seule.

21.8 Annexe H — Types d'erreur

Chaque exécution en échec porte un type d'erreur issu de la liste fermée ci-dessous (champ `error.type`). Cette liste reprend le tableau de référence du chapitre 17 (§ 17.5.2), qui détaille en outre la première vérification à effectuer pour chaque type. Le message détaillé, propre à chaque cas, figure dans le détail de l'exécution.

Type	Signification	Cause typique	État	Renvoi
`timeout`	Le délai maximal de l'exécution ou d'une attente a été dépassé	Ressource lente, flux filtré sans rejet	`CRITICAL`	§ 19.6
`dns_error`	Le nom n'a pas pu être résolu	Nom erroné, DNS injoignable	`CRITICAL`	§ 19.5
`connection_refused`	La connexion a été refusée	Service arrêté, mauvais port	`CRITICAL`	§ 19.6
`connection_reset`	La connexion a été interrompue par le serveur ou un équipement intermédiaire	Pare-feu, surcharge, plantage du service	`CRITICAL`	§ 19.6
`tls_error`	La négociation TLS a échoué	Autorité inconnue, nom de certificat incorrect, protocole obsolète	`CRITICAL`	§ 19.6
`certificate_expired`	Le certificat du serveur est expiré	Renouvellement oublié	`CRITICAL`	§ 19.6
`http_status`	Le code de réponse HTTP n'est pas celui attendu	Erreur serveur, redirection, authentification	`CRITICAL`	§ 19.6

Type	Signification	Cause typique	État	Renvoi
`assertion_failed`	Une assertion ou une règle n'est pas satisfaite	Contenu modifié, incident fonctionnel, valeur métier hors seuil	`CRITICAL` (`WARNING` pour une règle de sévérité `warning`)	§ 19.6, § 19.7
`element_not_found`	Un élément attendu n'a pas été trouvé dans la page	Évolution de la page, sélecteur fragile	`CRITICAL`	§ 19.7
`navigation_error`	Le navigateur n'a pas pu charger une page	Réseau, URL, certificat	`CRITICAL`	§ 19.7
`script_error`	Le scénario scripté a levé une erreur de programmation	Erreur dans le script	`CRITICAL`	§ 19.7
`browser_crash`	Chromium s'est arrêté anormalement	Mémoire insuffisante, page trop lourde	`UNKNOWN`	§ 19.7
`plugin_error`	Le plugin Nagios n'a pas pu s'exécuter jusqu'à son terme, ou a renvoyé un code retour en dehors de 0 à 3	Plugin absent ou non exécutable, arguments invalides, plugin interrompu	`UNKNOWN`	§ 19.6
`database_error`	La connexion ou la requête SQL a échoué	Paramètres, pilote, requête	`CRITICAL`	§ 19.6
`no_data`	Un contrôle `database` n'a renvoyé aucune ligne ; les règles ne peuvent pas être évaluées	Requête sans fonction d'agrégation, données absentes	`UNKNOWN` (par défaut, selon `onEmpty`)	§ 19.6, chapitre 11
`auth_failed`	L'authentification a été refusée	Mot de passe changé, compte verrouillé	`CRITICAL`	§ 19.6
`secret_missing`	Un secret référencé par le contrôle est absent de la sonde	Secret non créé, supprimé ou mal orthographié dans le contrôle ; configuration importée sans ses secrets	`UNKNOWN`	Chapitre 9, § 9.8.2 et § 9.12.3
`resource_exhausted`	La sonde a manqué de ressources	Mémoire, concurrence trop élevée	`UNKNOWN`	§ 19.6, § 19.7
`runner_crash`	Le moteur de scénarios s'est interrompu pendant l'exécution	Incident logiciel	`UNKNOWN`	§ 19.7
`agent_restart`	Le moteur principal a redémarré pendant l'exécution	Mise à jour, coupure, redémarrage	`UNKNOWN`	Chapitre 20
`maintenance`	L'exécution a été interrompue à l'expiration du délai de drainage, au début de l'activation d'une mise à jour	Exécution plus longue que `updates.drainTimeout`	`UNKNOWN`	Chapitre 20
`unknown`	Erreur non classée	Divers	`UNKNOWN`	§ 19.12

Information — L'état indiqué est l'état habituel. Le classement définitif d'un type d'erreur en `CRITICAL` ou en `UNKNOWN` est décrit au chapitre 8. Les textes exacts des messages d'erreur sont propres à chaque version du logiciel et figurent dans ses notes de version ; le type d'erreur est l'information à exploiter dans vos règles.

21.9 Annexe I — États et codes retour Nagios

21.9.1 États de la sonde

État	Signification	Valeur de `sva.check.status`	Code retour de `check_sva`
`OK`	Contrôle réussi, dans les seuils	0	0
`WARNING`	Contrôle réussi, mais seuil de performance dépassé, ou plugin ayant renvoyé 1	1	1
`CRITICAL`	Échec technique ou fonctionnel confirmé (après les éventuelles nouvelles tentatives)	2	2
`UNKNOWN`	Le contrôle n'a pas pu être évalué (problème de la sonde, défaillance du navigateur, plugin ayant renvoyé 3)	3	3
`STALE`	Aucun résultat depuis deux intervalles pour un contrôle actif. Calcul suspendu pendant l'état de maintenance et pendant un intervalle après la reprise	—	3
Sonde en état de maintenance	Activation d'une mise à jour en cours, du drainage au retour en service (`/health` : `MAINTENANCE`) ; ce n'est pas un état de contrôle	—	3, avec le texte `MAINTENANCE`

21.9.2 Convention des plugins Nagios

Code retour	État Nagios	État de la sonde
0	OK	`OK`
1	WARNING	`WARNING`
2	CRITICAL	`CRITICAL`
3	UNKNOWN	`UNKNOWN`
Autre code, ou plugin interrompu	—	`UNKNOWN` (`plugin_error`)

Sortie d'un plugin : une ligne de texte lisible, suivie facultativement du caractère `|` et des données de performance au format `label=valeur[unité];avertissement;critique;min;max`.

Exemple de sortie de `check_sva` :

```
CRITICAL - boutique-connexion : étape 'Vérifier l'accès à l'espace client' (assertion_failed) |
    duration=14.65s;8;;
```

TEXTE

21.10 Annexe J — Correspondance avec la génération historique

Cette annexe s'adresse aux clients qui utilisaient la génération précédente de la plateforme de supervision applicative. Le détail de la conversion des scénarios figure aux chapitres 13 à 16.

Génération historique	Nouvelle génération	Remarque
Serveur central obligatoire (réception, stockage, interface)	Sonde autonome : stockage local, interface d'administration, API	La console centrale devient facultative (évolution prévue)

Génération historique	Nouvelle génération	Remarque
Liaison WebSocket permanente vers le serveur	Aucune liaison obligatoire ; exports OTLP sortants facultatifs	—
États OK / WARNING / CRITICAL / STALE	`OK`, `WARNING`, `CRITICAL`, `UNKNOWN`, `STALE`	`UNKNOWN` ajouté pour distinguer les problèmes de la sonde
Scénarios WEB CasperJS + PhantomJS	Scénarios `browser` Puppeteer + Chromium (déclaratifs ou scriptés)	Réécriture nécessaire, intention fonctionnelle conservée
Enregistrement avec l'extension Resurrectio	Enregistreur de Chrome DevTools	Import des enregistrements : évolution prévue
`casper.waitForSelector`	`page.locator(…).wait()` ou `page.waitForSelector`	Attente explicite d'un élément
`casper.withFrame`	Propriété `frame` d'une étape, ou `contentFrame()`	—
Événements clavier via jQuery	`page.keyboard` ; action `press`	Plus de dépendance à jQuery
Captures dans les scripts	Capture automatique en cas d'échec	—
XPath absolus systématiques	Sélecteurs stables (`data-testid` , rôle, texte) ; XPath en dernier recours	—
Scénarios WS SoapUI (assertion « Contains »)	Contrôle `http` avec assertion de corps	Requête SOAP envoyée en `POST` XML. Import des projets SoapUI : évolution prévue
Projets SoapUI avancés	Contrôle `api` multi-requêtes avec extraction de variables	—
Scénarios BDD `[DATABASE]` / `[SQL]` / `[PROC]` / `[PARAM]` / `[RULES]`	Contrôle `database` : `query` ou `procedure` , `params` , `rules`	Opérateurs `LT` , `LTE` , `GT` , `GTE` , `EQ` conservés (`lt` … `eq`) ; mots de passe stockés comme secrets. Import automatique du format historique : évolution prévue
Plugins Nagios « en cours de développement »	Contrôle `nagios`	—
Remontée vers Nagios par NSCA (passif)	Plugin actif `check_sva` , ou intégration OpenTelemetry et Prometheus	—
Outil `scenarioRunner`	Exécution d'essai (`dryRun`) et lanceur local de scénarios	—
Rejeu d'un scénario depuis le serveur	`POST /api/v1/checks/{id}/runs`	—
Graphes de disponibilité et de performance	Interface d'administration (historique récent) et plateformes d'observabilité	Modèles Grafana et règles d'alerte Prometheus fournis, prêts à importer

→ **Évolution prévue** — Des outils d'import sont prévus dans une version ultérieure pour le format historique des contrôles de bases de données, les projets SoapUI et les enregistrements de l'enregistreur Chrome DevTools. La conversion se fait manuellement, à l'aide de cette annexe et des chapitres 11 et 13 à 16.

21.11 Annexe K — Informations de version et statut des informations

21.11.1 Identification

Élément	Valeur
Document	Manuel technique complet de la sonde de supervision SYNOVEX (SYN-DOC-00)
Produit	Gamme SYNOVEX : SYNOVEX Essential, SYNOVEX Professional, SYNOVEX Enterprise
Version de l'image système	Indiquée dans les notes de version et par `GET /api/v1/system` (champ `version`)
Versions de Java, OpenJDK, Node.js, `puppeteer-core`, Chromium et des pilotes JDBC	Indiquées dans les notes de version et par `GET /api/v1/system` (champ `versions`)
Fin de la période de support de sécurité	Mois et année indiqués à l'achat, selon les conditions de votre contrat, au moins 5 ans après la mise sur le marché de la version majeure ; exposée par `GET /api/v1/system` (champ `supportEndsAt`)
Éditeur	SYNOVEX — SIREN 884 713 009

21.11.2 Mises à jour du manuel

Ce manuel accompagne le logiciel SYNOVEX. Il est mis à jour lorsque le produit évolue ; les informations propres à chaque version du logiciel livrée (numéro, composants, changements) figurent dans ses notes de version et sont exposées par `GET /api/v1/system`.

21.11.3 Statut des informations

Ce manuel décrit le comportement de SYNOVEX sur Raspberry Pi 5. Pour l'exploiter correctement, tenez compte des conventions suivantes.

- **Exemples fictifs et domaines réservés** : les exemples s'appuient sur des domaines et des plages d'adresses réservés à la documentation, précisés ci-dessous ; remplacez-les par les valeurs de votre environnement.
- **Valeurs propres à chaque version** : les numéros de version du produit, de l'image système et des composants intégrés, la liste des plugins Nagios intégrés, la liste des SSD pris en charge et les textes des messages d'erreur sont indiqués dans les notes de version de chaque version livrée. Sur une sonde en service, `GET /api/v1/system` expose la version de l'image (champ `version`), celles des composants embarqués, dont Puppeteer et Chromium (champ `versions`), et la fin de la période de support (champ `supportEndsAt`).
- **Valeurs indicatives** : les valeurs présentées comme indicatives, notamment les limites de concurrence par défaut (annexe G, § 21.7.6), sont des points de départ à ajuster selon la charge observée sur votre sonde (chapitre 12, § 12.6).
- **Schémas d'API de référence** : l'annexe F résume les points d'accès de l'API REST ; la description complète des requêtes et des réponses, au format OpenAPI 3.1, est publiée par la sonde elle-même (`GET /api/v1/openapi.json`).

Les exemples (noms, adresses IP, domaines, comptes) sont **fictifs**. Ils utilisent les domaines `example.com`, `example.org`, `example.net`, `example.internal` et les plages d'adresses réservées à la documentation (`192.0.2.0/24`, `198.51.100.0/24`, `203.0.113.0/24`).

> **Information** — Le préfixe des métriques `sva.` (`sva_` au format Prometheus) est stable et indépendant du nom commercial du produit : les requêtes et tableaux de bord présentés dans ce manuel restent valables.

21.12 Annexe L — Assistance et contacts

Rubrique	Information
Service, adresse électronique, téléphone, horaires et langues du support	Indiqués dans votre contrat ; le portail de support (voir ce tableau) centralise les demandes dès sa mise en service
Site commercial	`https://synovex.incitypark.fr`
Portail de support	`https://support-synovex.incitypark.fr`, après l'annonce de sa mise en service (encadré ci-dessous)
Niveaux de service	Délais de prise en charge définis par votre contrat, par niveau de gravité (§ 21.12.1) et par édition
Garantie et remplacement de carte	Durée, conditions et procédure de retour selon vos conditions de vente ; procédure technique de remplacement : chapitre 20, § 20.6.4
Documentation en ligne et notes de version	`https://docs-synovex.incitypark.fr`, après l'annonce de sa mise en service (encadré ci-dessous)
Distribution des mises à jour	`https://updates-synovex.incitypark.fr` (utilisée par la sonde pour les mises à jour en ligne, après l'annonce de sa mise en service ; images signées)
Console de gestion centralisée	`https://fleet-synovex.incitypark.fr` : adresse réservée à la console centrale prévue ; non utilisée par la sonde
Signalement d'une vulnérabilité de sécurité	Canal dédié indiqué dans votre contrat, distinct des canaux ordinaires du support ; voir § 21.12.4
Service commercial (changement d'édition, reprise, remplacement de carte)	Coordonnées indiquées dans votre contrat ou votre bon de commande ; site commercial ci-dessus

> **Information** — Ces adresses constituent le plan de nommage officiel de l'éditeur ; les services correspondants sont mis en service progressivement. Une procédure qui s'appuie sur l'un de ces services (portail de support, documentation en ligne, téléchargement d'une image) ne s'applique qu'après l'annonce de sa mise en service par l'éditeur. Avant cette annonce, ou si un service ne répond pas, utilisez le canal de contact et de mise à disposition des images convenu dans votre contrat.

21.12.1 Niveaux de gravité proposés

Gravité	Définition	Exemple
1 — Bloquant	Sonde hors service, aucune supervision assurée	La sonde ne démarre plus
2 — Majeur	Fonction importante indisponible, sans contournement	Tous les scénarios navigateur en `UNKNOWN`
3 — Mineur	Fonction dégradée, contournement possible	Une destination d'export en erreur intermittente
4 — Demande	Question, demande d'information ou d'évolution	Conseil sur la conception d'un scénario

Les délais associés à chaque niveau sont définis par votre contrat.

21.12.2 Informations à préparer

Avant de contacter le support, rassemblez les informations listées au chapitre 19, § 19.12 : version, édition, numéro de série ou adresse MAC, description et horodatage du symptôme, identifiants des exécutions, actions déjà tentées. Ne transmettez jamais de secret.

21.12.3 Ce que couvrent l'assistance et la maintenance

Service	Accès	Référence
Correctifs et avis de sécurité de la version acquise, pendant la période de support	Toute sonde, sans contrat	Chapitre 20, § 20.4.6
Garantie et remplacement d'une carte défectueuse, sans rachat de l'édition acquise	Toute sonde, selon les conditions de vente	Chapitre 20, § 20.6.4
Documentation et notes de version	Toute sonde	Ci-dessus
Nouvelles versions fonctionnelles	Sondes couvertes par un contrat de maintenance	Chapitre 20, § 20.4.7
Assistance technique (diagnostic, conseil de configuration)	Sondes couvertes par un contrat de maintenance	Ce chapitre
Prestations : mise en service sur site, écriture de scénarios, formation, développement spécifique	Sur devis	Service commercial

Les conditions applicables, et notamment les délais de prise en charge, sont celles de vos conditions de vente et, le cas échéant, de votre contrat de maintenance. Aucun délai n'est garanti par cette documentation.

21.12.4 Signaler une vulnérabilité de sécurité

Si vous découvrez une vulnérabilité dans la sonde ou dans son logiciel intégré, signalez-la par le canal dédié indiqué dans votre contrat (tableau ci-dessus, rubrique « Signalement d'une vulnérabilité de sécurité »), et non par les canaux ordinaires du support. Votre contrat indique également où consulter la politique de divulgation coordonnée de l'éditeur, qui précise le traitement des signalements, ainsi que la clé de chiffrement à utiliser pour transmettre un signalement confidentiel.

Indiquez :

- la version et l'édition de la sonde ;
- la description de la vulnérabilité et son impact ;
- les étapes de reproduction.

Testez uniquement sur des sondes que vous possédez ou que vous êtes autorisé à tester, sans accéder aux données d'autrui ni perturber un service.

L'éditeur publie des avis de sécurité pour les vulnérabilités corrigées. Les correctifs sont diffusés par le canal `security` (chapitre 20, § 20.4).

21.13 Annexe M — Licences tierces et mentions

La sonde intègre des logiciels libres et des composants tiers, distribués selon leurs propres licences. Le tableau ci-dessous présente les principaux composants. **La liste complète des composants (inventaire SBOM), avec leurs versions et les textes complets des licences, est fournie avec chaque version du produit ; les notes de version indiquent où la consulter.**

Composant	Rôle	Licence
Raspberry Pi OS (base Debian)	Système d'exploitation	Ensemble de logiciels sous diverses licences libres (notamment GPL, LGPL, BSD, MIT) ; micrologiciels du Raspberry Pi soumis à la licence de leurs fournisseurs
Noyau Linux	Noyau du système	GPL-2.0

Composant	Rôle	Licence
OpenJDK	Environnement d'exécution Java du moteur principal	GPL-2.0 avec *Classpath Exception* ; distribution retenue indiquée dans l'inventaire des composants
Node.js	Environnement d'exécution du moteur de scénarios	MIT
Puppeteer (`puppeteer-core`)	Pilotage du navigateur	Apache-2.0
Chromium	Navigateur headless	BSD-3-Clause, avec de nombreux composants tiers sous leurs propres licences
SQLite	Base de données embarquée	Domaine public
Pilote JDBC SQLite	Accès à SQLite depuis Java	Pilote retenu et licence indiqués dans l'inventaire des composants
SDK OpenTelemetry pour Java	Production de la télémétrie	Apache-2.0
Monitoring Plugins (`monitoring-plugins`)	Plugins Nagios fournis	GPL-3.0
fping	Contrôles ICMP	Licence de type BSD
Pilote JDBC PostgreSQL	Contrôles `database`	BSD-2-Clause
Pilote JDBC MariaDB (MariaDB Connector/J)	Contrôles `database` MySQL et MariaDB	LGPL-2.1
Pilote JDBC Microsoft SQL Server	Contrôles `database`	MIT
Pilote JDBC Oracle	Contrôles `database`	Conditions d'utilisation gratuite d'Oracle (*Oracle Free Use Terms and Conditions*)
Autres bibliothèques	Moteur principal, moteur de scénarios, interface	Indiquées, avec leurs licences, dans l'inventaire des composants

Marques

Les marques citées dans cette documentation sont mentionnées dans les mentions légales, au début du document.

Mentions légales

Les mentions légales applicables à cette documentation figurent au début du document, dans les informations sur le document.