RÉFÉRENCE TECHNIQUE ET API

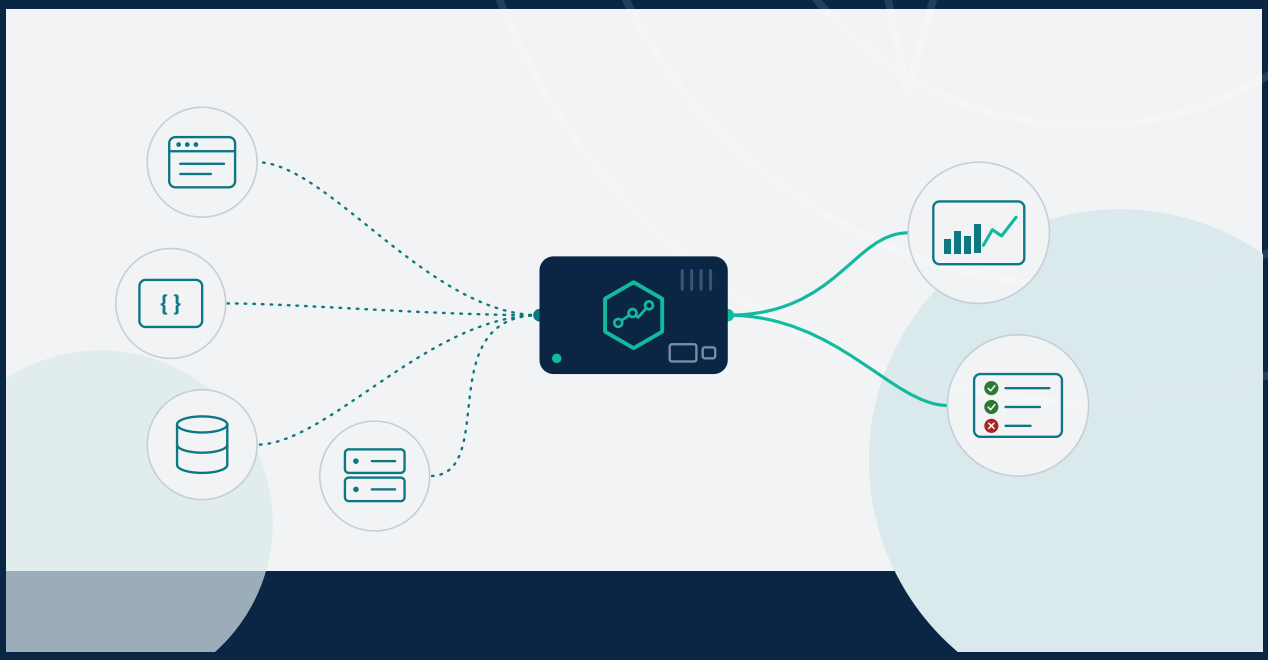
SYNOVEX

Référence technique et API

Matériel : Raspberry Pi 5

Éditions : Essential, Professional, Enterprise

Public : Intégrateurs avancés, architectes, développeurs, équipes chargées du développement du produit



SYNOVEX — SIREN 884 713 009

Table des matières

Informations sur le document 4

Chapitre 1 — Architecture de la solution 7
1.1 Principes d'architecture 7
1.2 Vue d'ensemble 7
1.3 Composants de la sonde 10
1.4 Stockage et conservation des données 13
1.5 Cycle de vie d'une exécution 13
1.6 Publication et interopérabilité des résultats 15
1.7 Administration de la sonde 16
1.8 Modes de fonctionnement 17
1.9 Sécurité intégrée 18
1.10 Comparaison avec l'architecture historique 19

Chapitre 2 — Éditions et fonctions 20
2.1 Périmètre de la solution 20
2.2 Édition de la sonde 21

Chapitre 3 — API REST 23
3.1 API REST : principes 23
3.2 Événements en temps réel 29
3.3 Référence de l'API REST 29

Chapitre 4 — Modèles de contrôles, de résultats et d'états 36
4.1 La notion de contrôle 36
4.2 Le résultat d'une exécution 39
4.3 Les états et les erreurs 41
4.4 Classes d'exécution et concurrence 45
4.5 Rétention et exploitation des données 46

Chapitre 5 — Modèles de planification 48
5.1 Planifier un contrôle 48
5.2 Les plans d'exécution 51
5.3 Comportement d'un plan 52

Chapitre 6 — Formats de configuration 57
6.1 Référence des configurations 57

Chapitre 7 — Télémétrie, événements et conventions de nommage 63
7.1 Conventions de télémétrie 63

Chapitre 8 — Types d'erreur, états et codes retour 70
8.1 Types d'erreur 70
8.2 États et codes retour Nagios 72

Chapitre 9 — Contraintes de sécurité et flux réseau 73
9.1 Principes de sécurité de la sonde 73
9.2 Flux réseau et pare-feu 74
9.3 Sécurisation des accès administrateur 76

Chapitre 10 — Glossaire 80
10.1 Glossaire 80

Chapitre 11 — Licences tierces et mentions 89
11.1 Licences tierces et mentions 89

Chapitre 12 — Schémas JSON 91
12.1 Réponse de GET /api/v1/checks/{id}/aggregates 91
12.2 Certificat TLS de la sonde 91
12.3 Réponse de GET /api/v1/checks/{id}/status 91
12.4 Contrôle 92
12.5 Définitions communes 93
12.6 Archive d'export de configuration 94
12.7 Résultat du test d'une destination d'export 95
12.8 Destination d'export 95
12.9 Réponse de GET /health 96
12.10 Plan d'exécution 96
12.11 Résultat d'exécution 96
12.12 Valeur d'un secret 97
12.13 Paramètres : section dot1x 98
12.14 Paramètres : section license (lecture seule) 98
12.15 Paramètres : section logging 99
12.16 Paramètres : section network 99
12.17 Paramètres : section proxy 100
12.18 Paramètres : section retention 100
12.19 Paramètres : section security 101
12.20 Paramètres : section ssh 101
12.21 Paramètres : section trust 102
12.22 Paramètres : section updates 102
12.23 Réponse de GET /api/v1/system 103
12.24 Jeton d'API 104
12.25 Réponse de GET /api/v1/updates 104
12.26 Compte de l'interface d'administration 106

SYNOVEX

Référence technique et API

Sonde de supervision applicative autonome pour Raspberry Pi 5

Information	Valeur
Document	Référence technique et API — identifiant SYN-DOC-06
Révision documentaire	B
Public	Intégrateurs avancés, architectes, développeurs, équipes chargées du développement du produit
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 au § 11.1 ; 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 (§ 8.13 du guide d'installation et d'administration).

À propos de ce guide

Cette référence rassemble les éléments techniques de SYNOVEX destinés aux intégrateurs avancés, aux architectes et aux équipes chargées du développement du produit : architecture et composants, éditions, API REST (principes, authentification, rôles, points d'accès, événements), modèles de contrôles, de résultats, d'états et de planification, formats de configuration, télémétrie et conventions de nommage, types d'erreur et codes retour, contraintes de sécurité, flux réseau, glossaire et structure des schémas JSON.

La description exhaustive des requêtes et des réponses de l'API est publiée par la sonde elle-même au format OpenAPI 3.1 (`GET /api/v1/openapi.json`). Publié par la sonde, ce document décrit l'API de la version installée. Pour le consulter hors ligne ou l'importer dans un outil (générateur de client, catalogue d'API), enregistrez-le depuis une sonde, par exemple avec la fonction `sapi` du § 3.1.3 : `sapi "$SONDE/api/v1/openapi.json" > openapi.json`. Le dernier chapitre présente la structure des schémas JSON de référence, propriété par propriété.

La bibliothèque documentaire SYNOVEX

La documentation SYNOVEX se compose d'un manuel technique complet et de six guides spécialisés. Tous sont produits à partir des mêmes sources : un passage repris dans plusieurs documents y est rigoureusement identique.

Document	Identifiant	Public
Manuel technique complet	SYN-DOC-00	Tous publics : référence intégrale
Guide de démarrage rapide	SYN-DOC-01	Techniciens d'installation, nouveaux clients
Guide d'installation et d'administration	SYN-DOC-02	Administrateurs système et réseau, équipes de sécurité
Guide d'utilisation et de supervision	SYN-DOC-03	Équipes d'exploitation et de supervision
Guide de développement des scénarios	SYN-DOC-04	Développeurs, intégrateurs, équipes de qualité
Guide d'intégration et d'observabilité	SYN-DOC-05	Architectes, DevOps, SRE
Référence technique et API	SYN-DOC-06	Intégrateurs avancés, architectes, développeurs

Renvois. Dans un guide, un renvoi sans autre mention désigne une section, un chapitre ou une figure du même guide, avec la numérotation de ce guide. Lorsque le sujet est traité dans un autre guide, le renvoi nomme ce guide, par exemple « § 6.3 du guide d'installation et d'administration ». La numérotation est alors celle du guide cité. Chaque guide se lit ainsi sans le manuel technique complet, qui réunit l'intégralité de la documentation sous sa propre numérotation.

Services en ligne. L'ensemble de la bibliothèque est également publié sur `https://docs-synovex.incitypark.fr`, avec une recherche et une navigation par profil. Ce site, le portail de support et le service de distribution des mises à jour de l'éditeur sont accessibles après l'annonce de leur mise en service. Avant cette annonce, ou si l'un de ces services ne répond pas, utilisez le canal de contact prévu par votre contrat. Les adresses citées dans ce guide s'entendent sous cette réserve.

Chapitre 1 — 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 9 et récapitulés en annexe (chapitre 10).

1.1 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

1.2 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 (§ 1.2.1). 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.

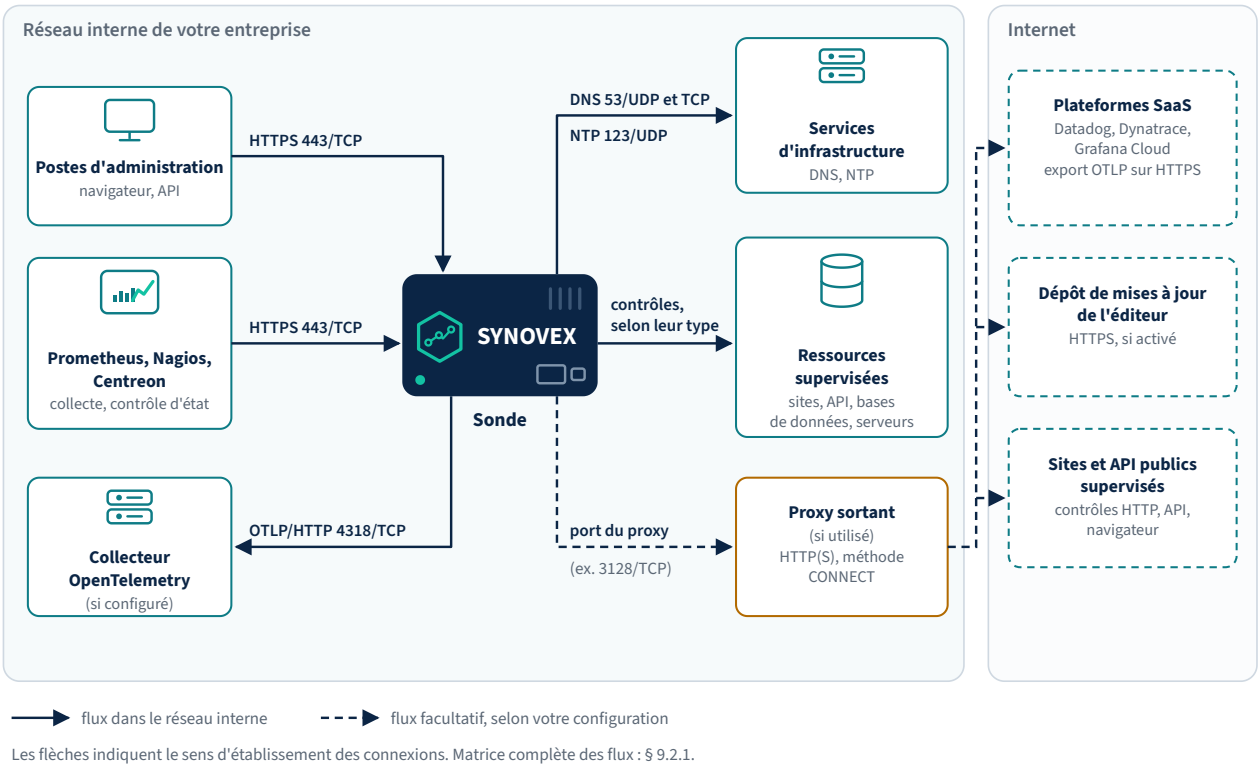


Figure 1-1 — Schéma d'implantation type : sonde dans le réseau interne, plateformes de supervision, accès Internet sortant via proxy

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.

1.2.1 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 4 du guide d'installation et d'administration), 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 2 du guide d'intégration et d'observabilité), 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 (§ 1.7). 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 (§ 2.12 du guide d'intégration et d'observabilité) ;
- **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.

1.3 Composants de la sonde

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

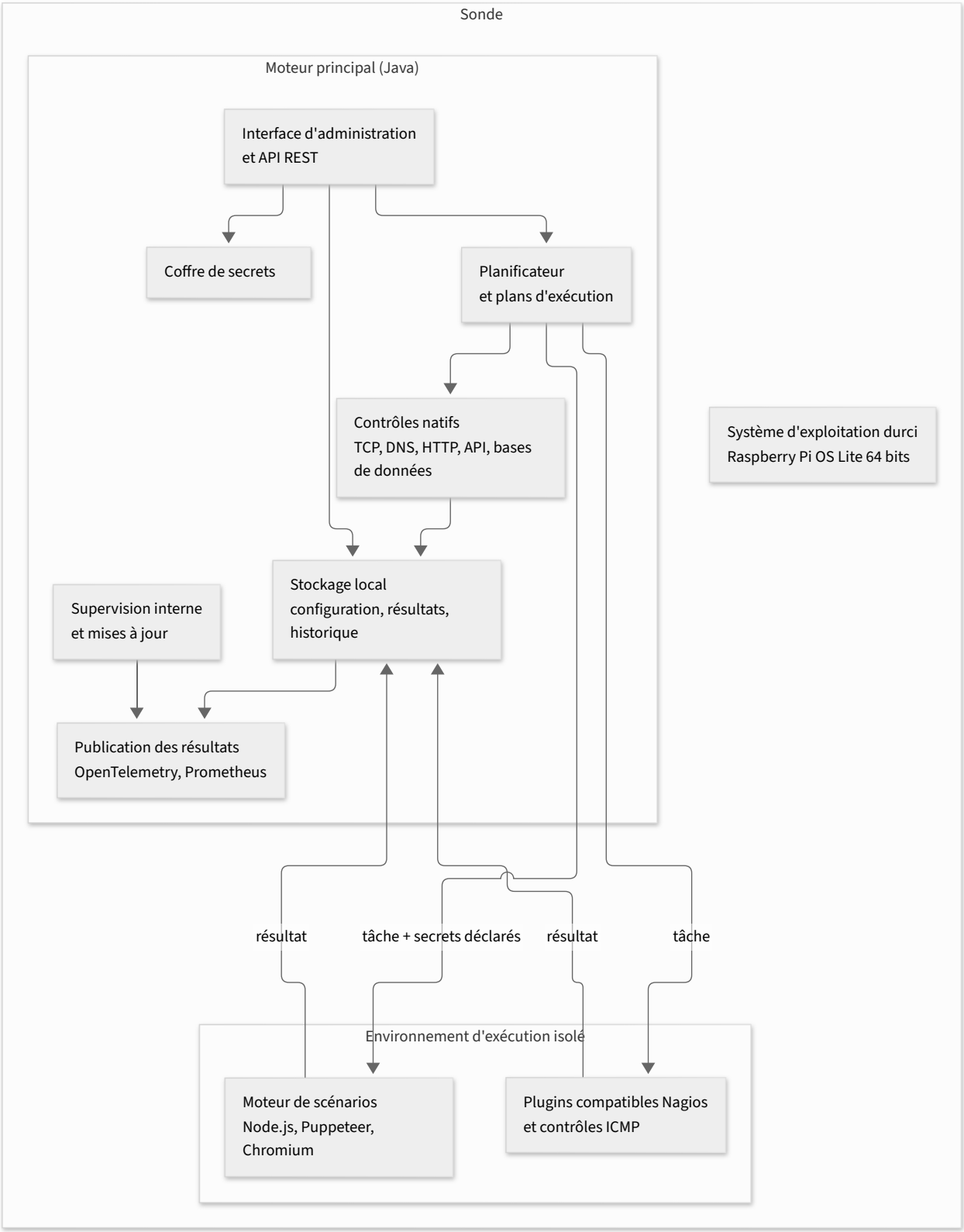


Schéma 1-1 — Composants de la sonde

1.3.1 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 2 du guide d'installation et d'administration). 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 8 du guide d'installation et d'administration).

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 (§ 8.3 du guide d'installation et d'administration). 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 (§ 8.4.5 du guide d'installation et d'administration). 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 (§ 8.3.4 du guide d'installation et d'administration).

1.3.2 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 3). 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 (§ 6.4 du guide d'installation et d'administration)
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 5)
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 5 du guide d'utilisation et de supervision)
Évaluation	Application des critères de réussite (assertions, seuils), des nouvelles tentatives et des règles de calcul de l'état (chapitre 4)
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 2 du guide d'intégration et d'observabilité)
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 (§ 8.4 du guide d'installation et d'administration)

1.3.3 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 3 du guide de développement des scénarios).

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 3 détaille les rôles.

1.3.4 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 5).

1.4 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 (§ 8.3.4 du guide d'installation et d'administration). Ces durées sont configurables dans les limites de l'espace disponible (chapitre 3). 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 8 du guide d'installation et d'administration détaille les mesures de protection de la carte microSD et les procédures de sauvegarde et de restauration.

1.5 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.

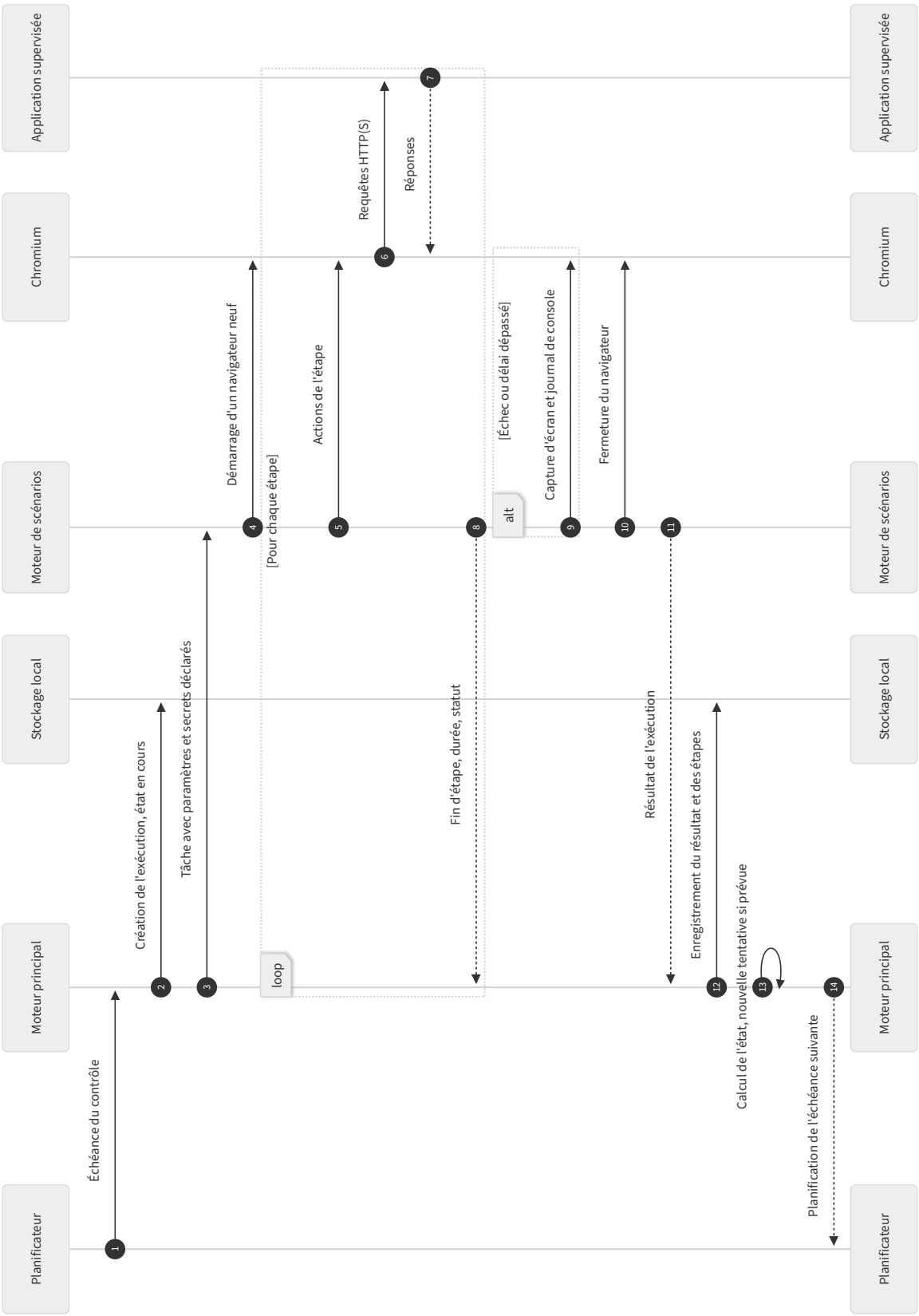


Schéma 1-2 — Cycle de vie d'une exécution (scénario navigateur)

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

1.6 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.

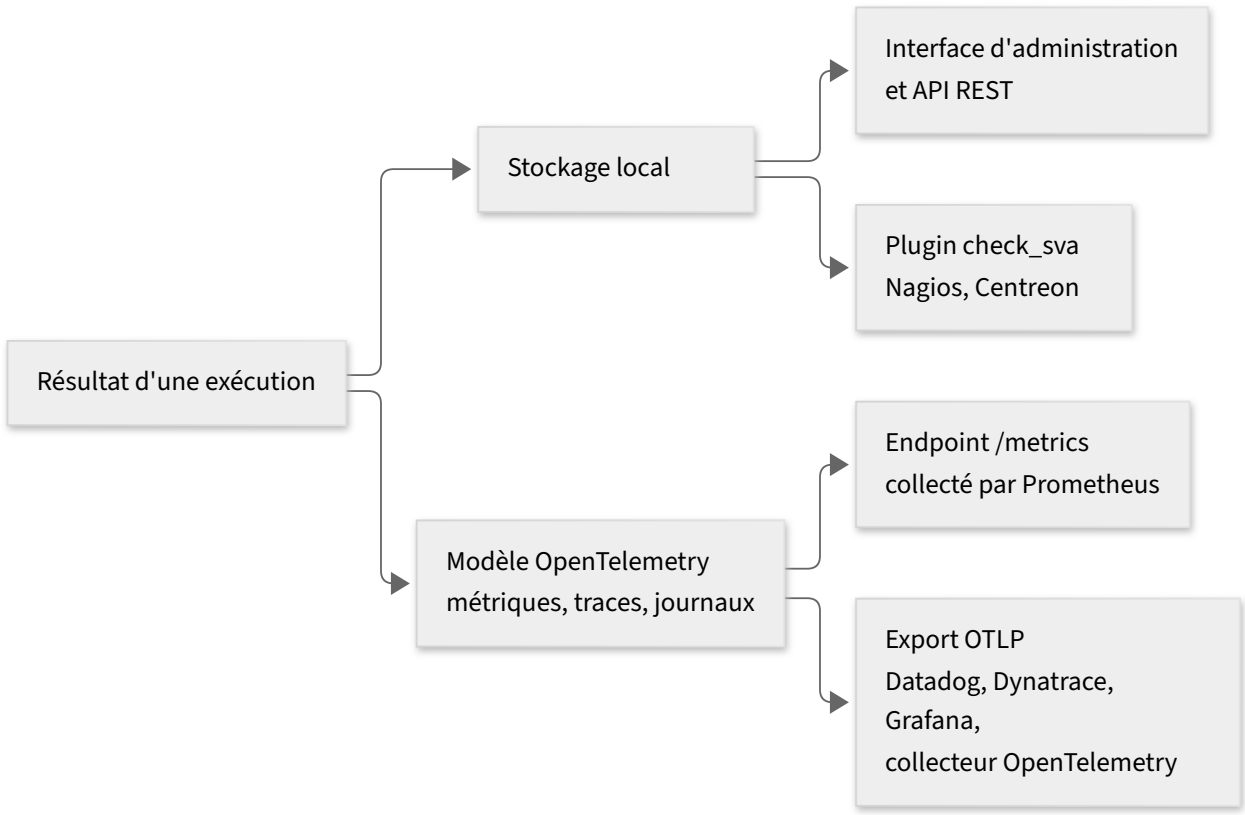


Schéma 1-3 — Publication des résultats : stockage local, API et exports

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 2 du guide d'intégration et d'observabilité 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 2 du guide d'intégration et d'observabilité.

1.7 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	chapitre 2 du guide d'installation et d'administration du guide d'installation et d'administration
Fichier de provisionnement	Configuration initiale sans intervention interactive, préparée sur la carte microSD avant le premier démarrage ; adapté aux déploiements en nombre	chapitre 2 du guide d'installation et d'administration du guide d'installation et d'administration
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	chapitre 1
API REST	Automatisation de toutes les opérations de l'interface : intégration à vos outils de déploiement, gestion en masse, synchronisation avec vos référentiels	chapitre 1 et chapitre 4 du guide d'utilisation et de supervision
Accès SSH	Maintenance avancée, désactivé par défaut et activable par un administrateur	chapitre 3 du guide d'installation et d'administration et chapitre 4 du guide d'installation et d'administration

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 3). 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.

1.8 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.

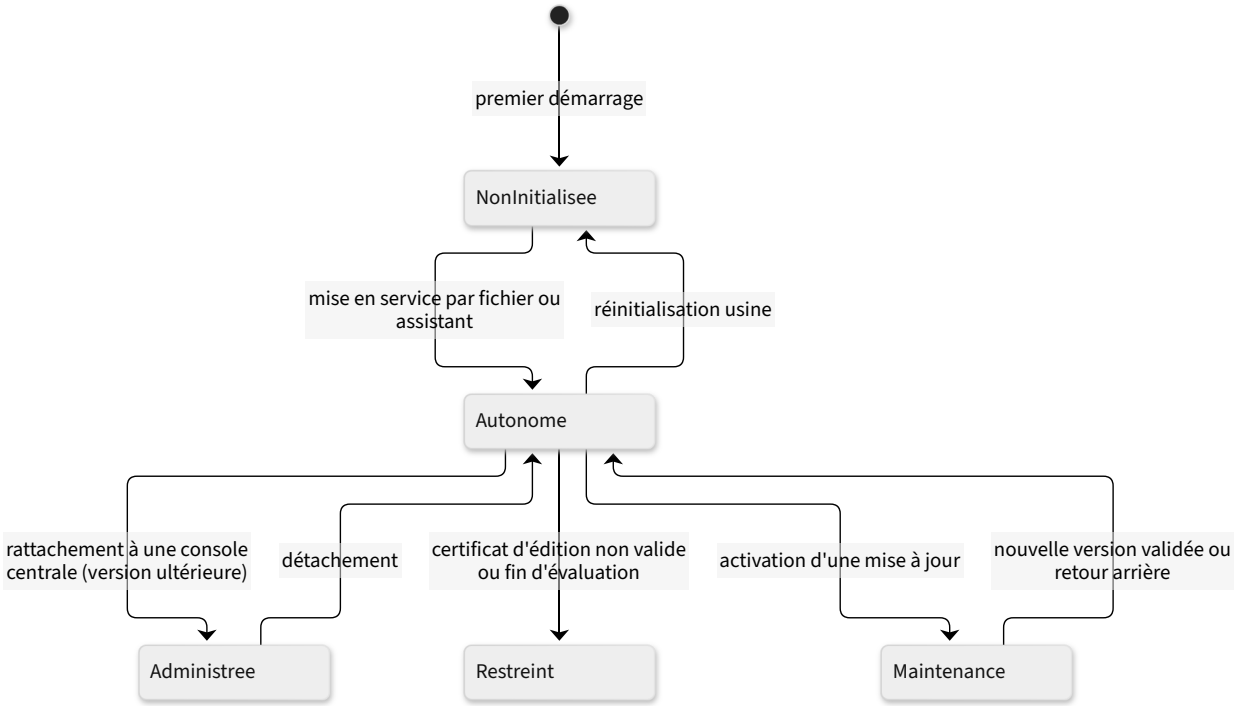


Schéma 1-4 — Modes de fonctionnement de la sonde

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 4 du guide d'installation et d'administration).
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.

Mode	Description
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.
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 (§ 8.4 du guide d'installation et d'administration).
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 (§ 2.2).

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 (§ 8.4 du guide d'installation et d'administration). 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 (§ 8.4.11 et § 8.4.12 du guide d'installation et d'administration). 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.

1.9 Sécurité intégrée

La sécurité est prise en compte à tous les niveaux. Le tableau ci-dessous résume les mesures, détaillées au chapitre 9 (réseau et sécurité) et au chapitre 3 (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

Domaine	Mesures
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 9)
Traçabilité	Journal d'audit des opérations d'administration, exportable vers votre outil de gestion des événements de sécurité
Mises à jour	Images système signées par l'éditeur, installées sur un emplacement A/B avec retour arrière automatique (§ 8.4.5 du guide d'installation et d'administration) ; 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 8 du guide d'installation et d'administration)

1.10 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 § 3.7 du guide de développement des scénarios donne la correspondance entre les anciennes pratiques et Puppeteer pour faciliter leur réécriture.

Chapitre 2 — Éditions et fonctions

2.1 Périmètre de la solution

2.1.1 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 (§ 8.4.7 du guide d'installation et d'administration). Pour passer à une édition supérieure, vous remplacez la carte par une carte de cette édition et vous y transférez votre configuration (§ 8.6 du guide d'installation et d'administration), sous réserve que le Raspberry Pi 5 corresponde aux exigences de l'édition visée (chapitre 2 du guide d'installation et d'administration).

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
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 5 (§ 6.6 du guide d'utilisation et de supervision), et le § 10.1 du guide d'installation et d'administration récapitule, sous forme d'aide-mémoire, les prérequis et la configuration matérielle minimale.

2.1.2 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

Offre	Contenu	Ce que vous fournissez
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 (§ 8.3.4 du guide d'installation et d'administration).

Le contenu exact de chaque offre et les références de matériel sont détaillés au chapitre 2 du guide d'installation et d'administration.

> **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** (§ 2.1.5 du guide d'installation et d'administration).

2.1.3 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 1 du guide d'utilisation et de supervision).

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 (§ 1.6.3 du guide d'utilisation et de supervision).

2.1.4 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.

2.2 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

Édition	Types de contrôles disponibles
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 (§ 10.1). 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 (§ 4.6.3 du guide d'installation et d'administration).

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 (§ 3.3.2 du guide d'installation et d'administration), 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 § 8.6 du guide d'installation et d'administration.

Chapitre 3 — API REST

3.1 API REST : principes

3.1.1 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`) (§ 8.4.9 du guide d'installation et d'administration)	Aucune
`GET /metrics`	Métriques au format Prometheus (chapitre 2 du guide d'intégration et d'observabilité)	Jeton de rôle `metrics`

3.1.2 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 (§ 6.4 du guide d'installation et d'administration). 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 (§ 6.14 du guide d'installation et d'administration).

3.1.3 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 (§ 6.5 du guide d'installation et d'administration) ; 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` (§ 4.4.4 du guide d'installation et d'administration), 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 (§ 6.5 du guide d'installation et d'administration), 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 § 7.9 du guide d'utilisation et de supervision.

> **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.

3.1.4 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
`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 (§ 6.4.2 du guide d'installation et d'administration)	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

Code	Signification	Cas typique
`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 (§ 3.1.6)
`412 Precondition Failed`	Condition de l'en-tête `If-Match` ou `If-None-Match` non satisfaite (§ 3.1.5)	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 (§ 3.1.5)	`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.

3.1.5 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}` (§ 6.8.4 du guide d'installation et d'administration) ou `GET /api/v1/users/{id}` (§ 6.4.2 du guide d'installation et d'administration), 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}`, § 6.8.2 du guide d'installation et d'administration), 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` (§ 6.4.2 du guide d'installation et d'administration), 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'
```

3.1.6 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

Règle	Conséquence pour vous
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 (§ 7.3.2 du guide d'utilisation et de supervision), et sous réserve de la règle suivante
La sonde ne fournit **pas d'instantané** : les exécutions purgées par la rétention (§ 4.5) 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"
```

3.2 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.

3.3 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"` (§ 3.1.3). 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 3.

3.3.1 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 (§ 3.1.5)
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 7 du guide d'utilisation et de supervision)
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

3.3.2 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
`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`

Méthode et chemin	Rôle minimal	Description
`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` (§ 6.6.1 du guide d'installation et d'administration)
`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
`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

Méthode et chemin	Rôle minimal	Description
`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 (§ 6.8 du guide d'installation et d'administration). 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}` (§ 6.8.4 du guide d'installation et d'administration)
`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 (§ 6.4.2 du guide d'installation et d'administration)
`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 (§ 6.4.2 du guide d'installation et d'administration)
`DELETE /api/v1/users/{id}`	`admin`	Suppression d'un compte ; ses sessions et ses jetons sont révoqués immédiatement

Méthode et chemin	Rôle minimal	Description
`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)
`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)

Méthode et chemin	Rôle minimal	Description
`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` (§ 8.4.2 et § 8.4.12 du guide d'installation et d'administration)
`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 (§ 4.4.6 du guide d'installation et d'administration)
`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 (§ 4.7.10 du guide d'utilisation et de supervision)
`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)
`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

3.3.3 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

Chapitre 4 — Modèles de contrôles, de résultats et d'états

4.1 La notion de contrôle

4.1.1 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.

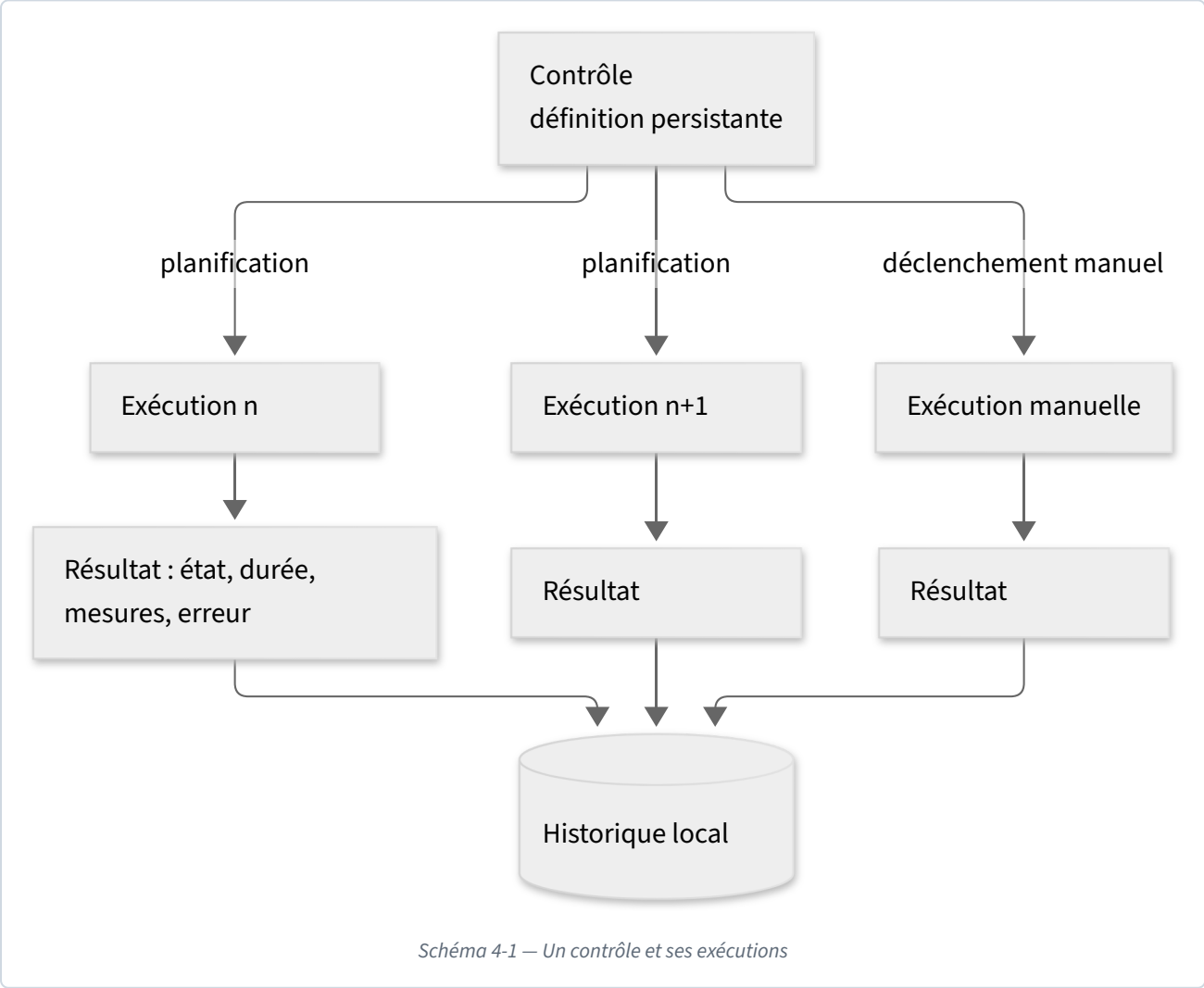


Schéma 4-1 — Un contrôle et ses exécutions

Les contrôles sont stockés **sur la sonde elle-même**. Ils continuent d'être exécutés quand aucune plateforme externe n'est joignable (§ 3.14 du guide d'utilisation et de supervision).

4.1.2 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	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`tcp`	L'ouverture d'un port TCP et, en option, un échange simple	Essential	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`dns`	La résolution d'un nom et la conformité de la réponse	Essential	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`http`	Une requête HTTP ou HTTPS : code de retour, en-têtes, contenu, durée, certificat	Essential	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`nagios`	L'exécution d'un plugin compatible Nagios	Essential (plugins fournis), Professional (plugins personnalisés)	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`api`	Une suite de requêtes HTTP enchaînées, avec réutilisation de valeurs d'une réponse à l'autre	Professional	chapitre 3 du guide d'installation et d'administration du guide d'utilisation et de supervision
`database`	Le résultat d'une requête ou d'une procédure SQL au regard de règles	Professional	chapitre 3 du guide d'installation et d'administration du guide d'utilisation et de supervision
`browser`	Un parcours utilisateur dans un navigateur Chromium sans interface (headless)	Enterprise	chapitres 1 à 4 du guide d'installation et d'administration du guide de développement des scénarios

4.1.3 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 (§ 4.1.2)	`http`
`enabled`	Contrôle actif ou suspendu	`true`
`tags`	Étiquettes libres (8 au maximum) servant au filtrage et au regroupement	`{ "application": "portail" }`
`schedule`	Planification (§ 3.3 du guide d'utilisation et de supervision et chapitre 5)	`{ "interval": "1m" }`

Propriété	Rôle	Exemple
`timeout`	Durée maximale d'une exécution (§ 3.4 du guide d'utilisation et de supervision)	`15s`
`retry`	Nouvelles tentatives avant de confirmer un échec (§ 3.5 du guide d'utilisation et de supervision)	`{ "attempts": 1, "interval": "20s" }`
`thresholds`	Seuils de performance (§ 3.6 du guide d'utilisation et de supervision)	`{ "warningDuration": "3s" }`
`secrets`	Liste des secrets que le contrôle est autorisé à utiliser (chapitre 3)	`["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 9) **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 9). Le comportement de chaque type de contrôle vis-à-vis d'IPv6 est précisé aux chapitres 4 et 5 du guide d'utilisation et de supervision (§ 4.1.3 du guide d'utilisation et de supervision).

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.

4.1.4 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 3 explique comment créer ce contrôle par l'interface d'administration ou par l'API, et le chapitre 4 du guide d'utilisation et de supervision détaille chacune des options du type `http`.

4.2 Le résultat d'une exécution

4.2.1 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 2 du guide d'intégration et d'observabilité).

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 (§ 3.5 du guide d'utilisation et de supervision)	`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 (§ 4.3.5), 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

Élément	Contenu	Exemple
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`
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…`

4.2.2 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` (§ 4.3.1 du guide de développement des scénarios), repris à l'identique au § 7.4 du guide d'utilisation et de supervision :

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 7 du guide d'utilisation et de supervision 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.

4.3 Les états et les erreurs

4.3.1 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 5 du guide d'utilisation et de supervision) ; certificat proche de l'expiration (chapitre 4 du guide d'utilisation et de supervision) ; 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 (§ 3.14.2 du guide d'utilisation et de supervision)

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 9 du guide d'installation et d'administration).

4.3.2 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 (§ 3.5 du guide d'utilisation et de supervision), 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`.

4.3.3 Cycle de vie de l'état d'un contrôle

Outre les cinq états définis au § 4.3.1, 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 (§ 6.6.5 du guide d'installation et d'administration) ;
- **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`.

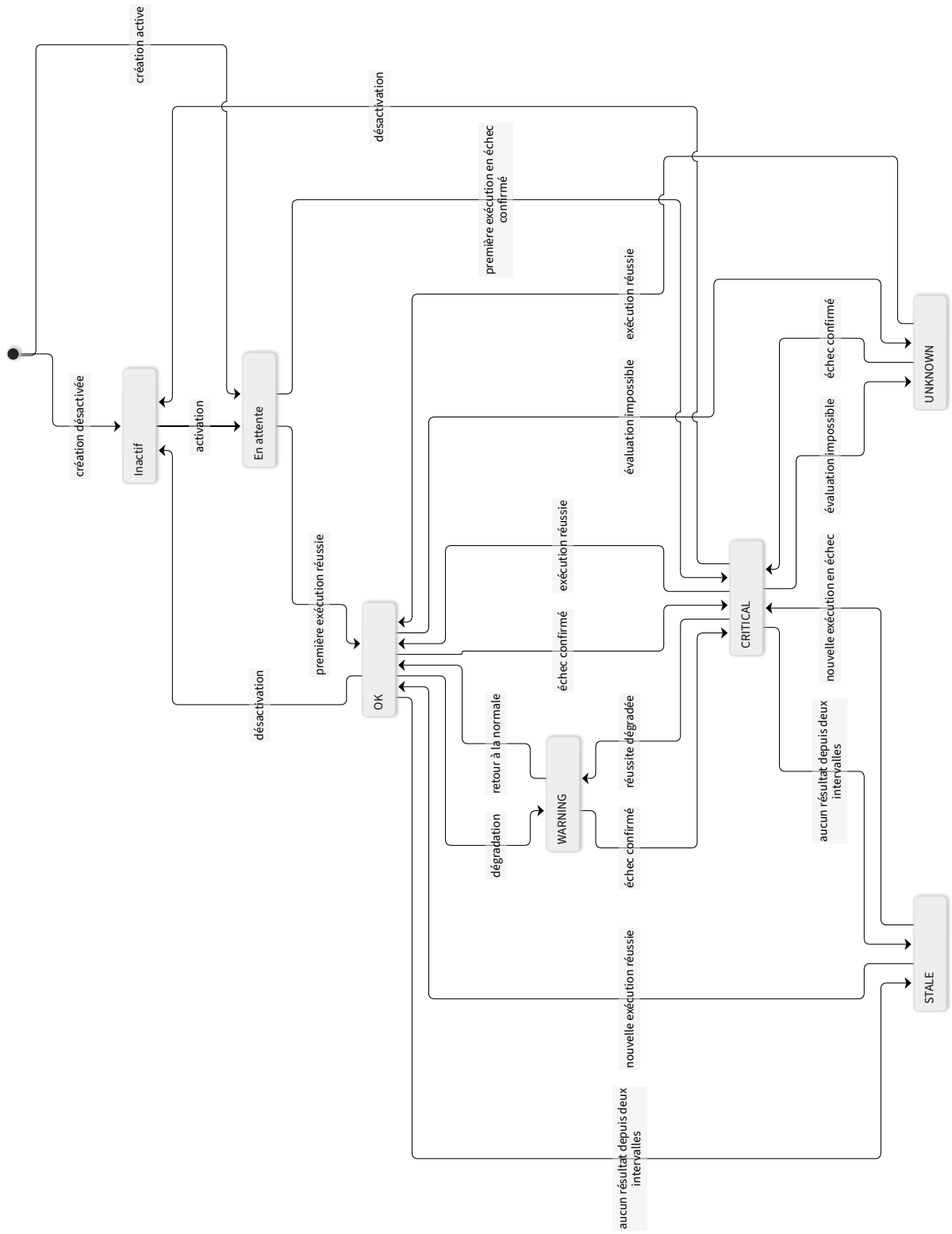


Schéma 4-2 — Cycle de vie de l'état d'un contrôle

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 (§ 3.2) et exporté sous forme de log (chapitre 2 du guide d'intégration et d'observabilité).

4.3.4 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 2 du guide d'intégration et d'observabilité 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 (§ 3.14.2 du guide d'utilisation et de supervision) 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` (chapitre 7 du guide d'utilisation et de supervision et chapitre 2 du guide d'intégration et d'observabilité). 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.

4.3.5 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 § 3.10 du guide d'utilisation et de supervision :

- **`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, § 4.3.2), `secret_missing` (secret référencé par le contrôle mais absent de la sonde), `resource_exhausted`, `runner_crash`, `agent_restart`, `maintenance` (§ 3.14.2 du guide d'utilisation et de supervision), `unknown`, et `no_data` par défaut (paramètre `onEmpty`, chapitre 5 du guide d'utilisation et de supervision).

Le tableau de référence de tous les types d'erreur, avec leur signification, leur origine probable et la première vérification à mener, figure au § 7.5.2 du guide d'utilisation et de supervision. Le chapitre 9 du guide d'installation et d'administration les reprend sous forme de fiches de diagnostic.

4.4 Classes d'exécution et concurrence

4.4.1 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.

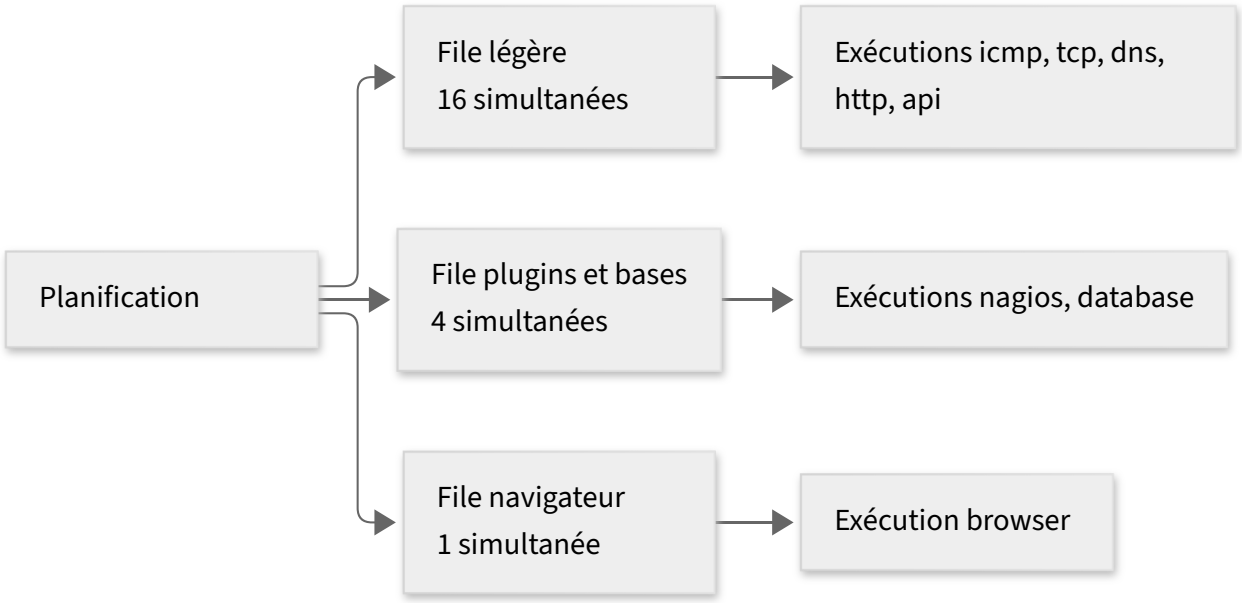


Schéma 4-3 — Files d'exécution et concurrence

4.4.2 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 2 du guide d'intégration et d'observabilité).

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 5 (§ 6.6 du guide d'utilisation et de supervision) 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.

4.5 Rétention et exploitation des données

4.5.1 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 3)

Ces durées sont configurables par un administrateur (§ 6.9 du guide d'installation et d'administration), 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 (§ 3.5.2 du guide d'utilisation et de supervision).

> **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.

4.5.2 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 2 et 8 du guide d'installation et d'administration (§ 8.3.4 du guide d'installation et d'administration).

4.5.3 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	chapitre 1 et chapitre 9 du guide d'utilisation et de supervision
Tableaux de bord, alertes, historique de long terme, vue multi-sondes	Plateformes d'observabilité via OpenTelemetry ou Prometheus	chapitre 1 du guide d'utilisation et de supervision du guide d'intégration et d'observabilité
Intégration aux outils de supervision existants	Plugin `check_sva` pour Nagios et Centreon	chapitre 1 du guide d'utilisation et de supervision du guide d'intégration et d'observabilité

La sonde reste dans tous les cas la **source de référence** des résultats : les plateformes externes en reçoivent une copie.

Chapitre 5 — Modèles de planification

5.1 Planifier un contrôle

5.1.1 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 (§ 3.3.2 du guide d'utilisation et de supervision)	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 2 du guide d'intégration et d'observabilité).

5.1.2 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

Expression	Signification
`*/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 (§ 5.3.4).

Une gigue (`jitter`) peut aussi s'appliquer à une planification cron. Elle évite que tous les contrôles planifiés « à l'heure pile » démarrent simultanément.

5.1.3 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 (§ 5.3.4) et l'heure de début des fenêtres de maintenance des mises à jour (chapitre 8 du guide d'installation et d'administration) 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

Situation	Effet
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 (§ 8.4.8 du guide d'installation et d'administration)
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` (§ 6.9 du guide d'installation et d'administration).

5.1.4 Choisir entre intervalle et cron

Besoin	Mode recommandé
Surveiller en continu la disponibilité d'un service	Intervalle
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

5.1.5 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

Type	Intervalle conseillé	Gigue conseillée	Timeout conseillé
`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.

5.1.6 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" }
  }
}
```

5.2 Les plans d'exécution

5.2.1 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

5.2.2 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` (§ 5.1)	—
`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` (§ 5.3.2).

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.

5.3 Comportement d'un plan

5.3.1 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` (§ 5.3.2) ;
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 (§ 5.3.3).

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.

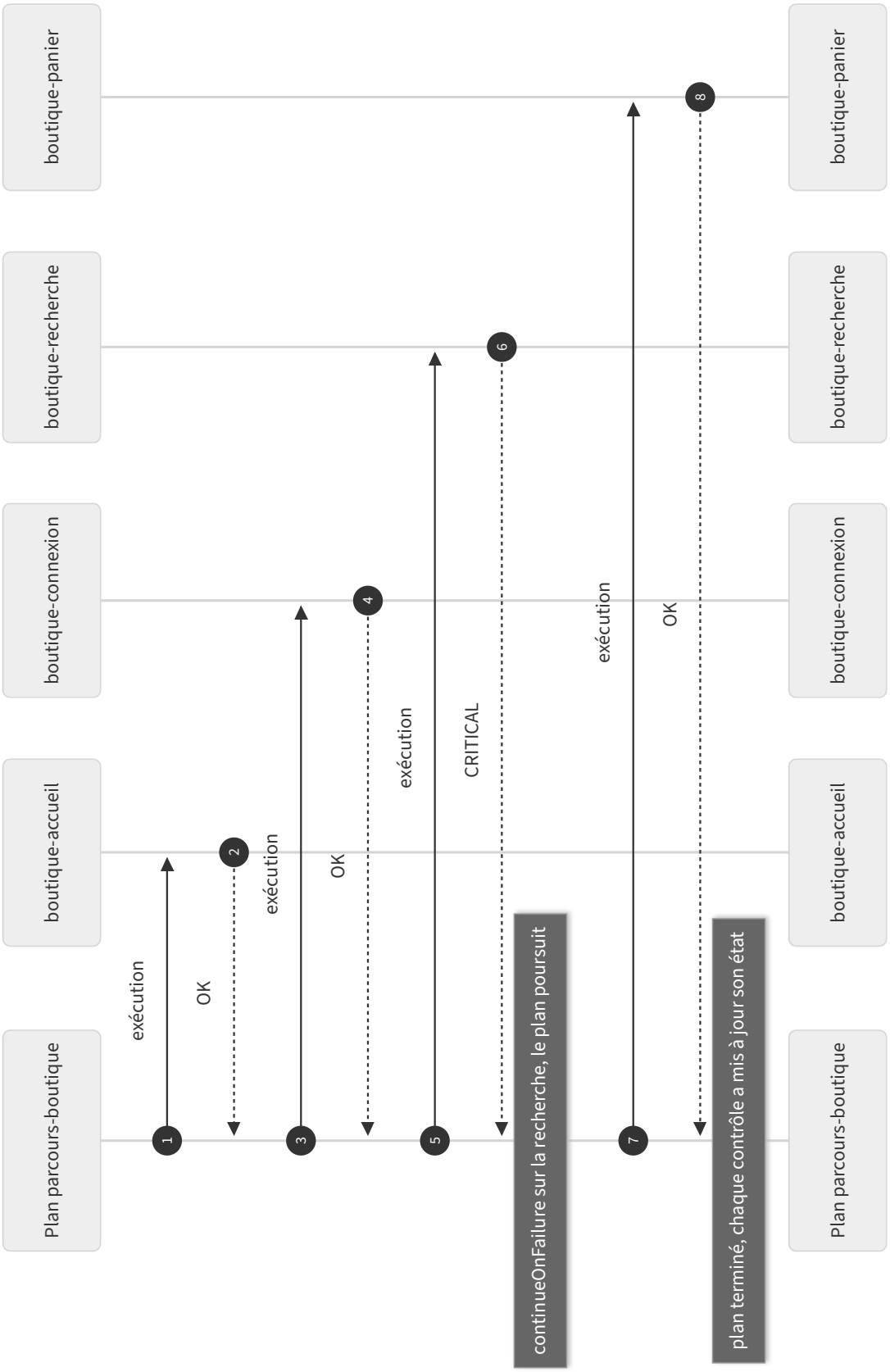


Schéma 5-1 — Déroulement d'un plan d'exécution

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

5.3.2 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` (§ 5.3.5). C'est le comportement attendu : il indique que cette partie du parcours n'a pas pu être évaluée.

5.3.3 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 (§ 6.6 du guide d'utilisation et de supervision).

5.3.4 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`

Propriété	Valeurs
`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 § 5.1.3.

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.

5.3.5 Plans, états et STALE

Un contrôle est déclaré `STALE` lorsqu'il ne produit aucun résultat pendant plus de deux intervalles de sa planification (§ 4.3.4). 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.

5.3.6 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.

Chapitre 6 — Formats de configuration

6.1 Référence des configurations

Les tableaux ci-dessous résument les champs. Les schémas complets et les exemples figurent dans les chapitres indiqués.

6.1.1 Champs communs à tous les contrôles (chapitre 4)

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`
`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}`.

6.1.2 Champs propres à chaque type

Type	Champs principaux	Chapitre
`icmp`	`host`, `count` (1 à 20), `maxPacketLossPercent`, `warningRttMs`, `criticalRttMs`	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`tcp`	`host`, `port`, `tls`, `send`, `expect`	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision

Type	Champs principaux	Chapitre
`dns`	`name`, `recordType` (`A`, `AAAA`, `CNAME`, `MX`, `TXT`, `NS`, `SRV`, `PTR`, `SOA`), `servers`, `expect`	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`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`)	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`api`	`requests` (1 à 20) : champs de `http` + `name`, `extract` (`type` : `jsonpath`, `xpath`, `regex` ; `reveal` pour afficher la valeur, masquée par défaut)	chapitre 3 du guide d'installation et d'administration du guide d'utilisation et de supervision
`nagios`	`plugin` (nom sans chemin), `args`, `perfdataExport`	chapitre 2 du guide d'installation et d'administration du guide d'utilisation et de supervision
`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)	chapitre 3 du guide d'installation et d'administration du guide d'utilisation et de supervision
`browser`	`mode` (`steps` ou `script`), `steps` (50 maximum), `scriptRef`, `params`, `viewport`, `userAgentSuffix`, `captureOnFailure`, `blockDomains`	chapitre 2 du guide d'installation et d'administration du guide de développement des scénarios

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`.

6.1.3 Plan d'exécution (chapitre 5)

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`

6.1.4 Destination d'export

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`

6.1.5 Fichier de provisionnement `provision.yaml`

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` (§ 6.1.7), **aucune fenêtre par défaut**
`proxy.url`, `proxy.noProxy`	Proxy sortant et exclusions

6.1.6 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)

Paramètre	Valeur par défaut
Page de récupération du compte administrateur	Ouverte 30 minutes après le démarrage ; 3 essais au plus par ouverture
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

6.1.7 Sections de paramètres système : réseau IPv6, 802.1X, SSH et mises à jour

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 (§ 8.4.2 du guide d'installation et d'administration)
`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 (§ 8.4.11 du guide d'installation et d'administration)
`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 : § 8.4.2 et § 8.4.11 du guide d'installation et d'administration)
`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.

Chapitre 7 — Télémétrie, événements et conventions de nommage

7.1 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.

7.1.1 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 4 du guide d'installation et d'administration). 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).

7.1.2 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
`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 5 du guide d'utilisation et de supervision)
`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

Métrique	Instrument	Unité	Attributs	Usage
`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 8 du guide d'installation et d'administration)
`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 9). 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 (§ 8.4 du guide d'installation et d'administration). Leur exploitation dans les plateformes d'alerte est décrite au § 2.15 du guide d'intégration et d'observabilité.

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 (§ 8.4.12 du guide d'installation et d'administration)

Métrique	Instrument	Unité	Attributs	Usage
`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

7.1.3 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`
`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`

Nom OpenTelemetry	Séries Prometheus correspondantes
`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` (§ 2.5.2 du guide d'intégration et d'observabilité) avant d'écrire vos règles d'alerte.

7.1.4 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 (§ 7.5.2 du guide d'utilisation et de supervision), dont `maintenance` pour une exécution interrompue par le drainage d'une maintenance (§ 2.15.1 du guide d'intégration et d'observabilité)

Attribut	Nombre de valeurs	Garde-fou
`sva.step.name`	50 au maximum par scénario	Noms d'étapes fixes, imposés par le contrat des scénarios (chapitre 4 du guide de développement des scénarios)
`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 (§ 2.4.3 du guide d'intégration et d'observabilité), à `true` par défaut, est alors positionné à `false`.

7.1.5 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 7 du guide d'utilisation et de supervision). 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.

7.1.6 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

Événement	Sévérité	Contenu
`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 (§ 2.11 du guide d'intégration et d'observabilité)
`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 § 8.4.2 du guide d'installation et d'administration) : 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 § 8.4.9 du guide d'installation et d'administration.

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.

Chapitre 8 — Types d'erreur, états et codes retour

8.1 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 § 7.5.2 du guide d'utilisation et de supervision, 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`	§ 8.2 du guide d'utilisation et de supervision
`dns_error`	Le nom n'a pas pu être résolu	Nom erroné, DNS injoignable	`CRITICAL`	§ 9.5 du guide d'installation et d'administration
`connection_refused`	La connexion a été refusée	Service arrêté, mauvais port	`CRITICAL`	§ 8.2 du guide d'utilisation et de supervision
`connection_reset`	La connexion a été interrompue par le serveur ou un équipement intermédiaire	Pare-feu, surcharge, plantage du service	`CRITICAL`	§ 8.2 du guide d'utilisation et de supervision
`tls_error`	La négociation TLS a échoué	Autorité inconnue, nom de certificat incorrect, protocole obsolète	`CRITICAL`	§ 8.2 du guide d'utilisation et de supervision
`certificate_expired`	Le certificat du serveur est expiré	Renouvellement oublié	`CRITICAL`	§ 8.2 du guide d'utilisation et de supervision
`http_status`	Le code de réponse HTTP n'est pas celui attendu	Erreur serveur, redirection, authentification	`CRITICAL`	§ 8.2 du guide d'utilisation et de supervision
`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`)	§ 8.2 du guide d'utilisation et de supervision, § 10.3 du guide de développement des scénarios
`element_not_found`	Un élément attendu n'a pas été trouvé dans la page	Évolution de la page, sélecteur fragile	`CRITICAL`	§ 10.3 du guide de développement des scénarios
`navigation_error`	Le navigateur n'a pas pu charger une page	Réseau, URL, certificat	`CRITICAL`	§ 10.3 du guide de développement des scénarios
`script_error`	Le scénario scripté a levé une erreur de programmation	Erreur dans le script	`CRITICAL`	§ 10.3 du guide de développement des scénarios

Type	Signification	Cause typique	État	Renvoi
`browser_crash`	Chromium s'est arrêté anormalement	Mémoire insuffisante, page trop lourde	`UNKNOWN`	§ 10.3 du guide de développement des scénarios
`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`	§ 8.2 du guide d'utilisation et de supervision
`database_error`	La connexion ou la requête SQL a échoué	Paramètres, pilote, requête	`CRITICAL`	§ 8.2 du guide d'utilisation et de supervision
`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`)	§ 8.2 du guide d'utilisation et de supervision, chapitre 5 du guide d'utilisation et de supervision
`auth_failed`	L'authentification a été refusée	Mot de passe changé, compte verrouillé	`CRITICAL`	§ 8.2 du guide d'utilisation et de supervision
`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`	§ 6.8.2 et § 6.12.3 du guide d'installation et d'administration
`resource_exhausted`	La sonde a manqué de ressources	Mémoire, concurrence trop élevée	`UNKNOWN`	§ 8.2 du guide d'utilisation et de supervision, § 10.3 du guide de développement des scénarios
`runner_crash`	Le moteur de scénarios s'est interrompu pendant l'exécution	Incident logiciel	`UNKNOWN`	§ 10.3 du guide de développement des scénarios
`agent_restart`	Le moteur principal a redémarré pendant l'exécution	Mise à jour, coupure, redémarrage	`UNKNOWN`	Chapitre 8 du guide d'installation et d'administration
`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 8 du guide d'installation et d'administration
`unknown`	Erreur non classée	Divers	`UNKNOWN`	§ 9.9 du guide d'installation et d'administration

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 4. 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.

8.2 États et codes retour Nagios

8.2.1 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`

8.2.2 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;;
```

Chapitre 9 — Contraintes de sécurité et flux réseau

9.1 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 (§ 4.2 du guide d'installation et d'administration)
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 (§ 9.2.1)
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 (§ 9.2.1)
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.

9.2 Flux réseau et pare-feu

9.2.1 Matrice des flux

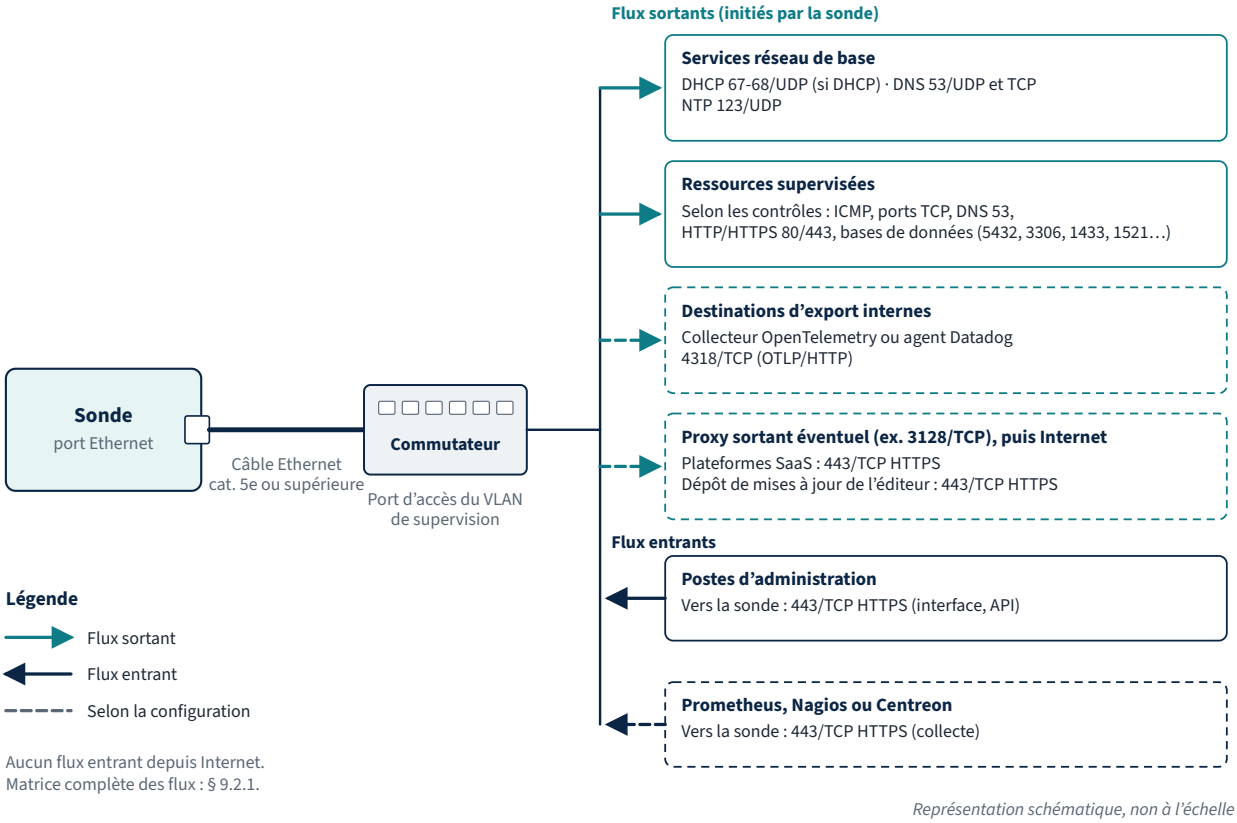


Figure 9-1 — Raccordement réseau de la sonde et principaux flux (le détail figure dans la matrice des flux ci-dessous)

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 (§ 9.3.4)
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 (§ 5.2.6 du guide d'installation et d'administration)

Sens	Source	Destination	Port / protocole	Usage	Obligatoire
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 (§ 5.2.5 du guide d'installation et d'administration)
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 (§ 5.5 du guide d'installation et d'administration)	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 (§ 2.1.4 du guide d'intégration et d'observabilité)	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 (§ 2.5.3 du guide d'intégration et d'observabilité)	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 (§ 6.14 du guide d'installation et d'administration) utilisera une connexion sortante initiée par la sonde, sans ouverture entrante. Aucun flux vers elle n'est nécessaire aujourd'hui.

9.2.2 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 9 du guide d'installation et d'administration).

9.2.3 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, § 5.7.1 du guide d'installation et d'administration), 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 (§ 5.2.5 du guide d'installation et d'administration).

> → **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.

9.3 Sécurisation des accès administrateur

9.3.1 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)
`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 3. 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 (§ 6.4.2 du guide d'installation et d'administration).

9.3.2 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.

9.3.3 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 (§ 9.2.2) et liste d'adresses autorisées (§ 9.2.3)
Poste d'administration	Utiliser un poste maîtrisé, à jour, de préférence un poste d'administration dédié

9.3.4 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`, § 5.2.4 du guide d'installation et d'administration). 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 (§ 9.2.3) 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 § 9.2.4 du guide d'installation et d'administration.

> **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 (§ 8.10 du guide d'installation et d'administration).

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` (§ 4.3.3 du guide d'installation et d'administration) ; 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) : § 3.1.3
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 9 du guide d'installation et d'administration). La récupération du compte administrateur par code de récupération reste possible (chapitre 8 du guide d'installation et d'administration).

Chapitre 10 — Glossaire

10.1 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 9
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 8 du guide d'installation et d'administration
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 2 du guide d'intégration et d'observabilité
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
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 3 et § 3.3
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 1 du guide d'utilisation et de supervision
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 4 du guide d'installation et d'administration
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 chapitre 6 du guide de développement des scénarios et chapitre 9 du guide d'installation et d'administration
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 3 et 7 du guide d'utilisation et de supervision

Terme	Définition
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 6 du guide de développement des scénarios
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 § 2.2
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 § 2.2
`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 2 du guide d'intégration et d'observabilité
Chrome DevTools Protocol (CDP)	Interface de pilotage de Chromium, utilisée par Puppeteer pour commander le navigateur. Voir chapitre 3 du guide de développement des scénarios
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 4
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 4 et 8 du guide d'installation et d'administration
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 4 et 8 du guide d'installation et d'administration
Collecte (*scrape*)	Interrogation périodique de l'adresse `/metrics` de la sonde par Prometheus (mode *pull*). Voir chapitre 2 du guide d'intégration et d'observabilité
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 2 du guide d'intégration et d'observabilité

Terme	Définition
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 9
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 6 du guide de développement des scénarios
Datadog	Plateforme d'observabilité en mode SaaS. Elle reçoit les données de la sonde via OTLP. Voir chapitre 2 du guide d'intégration et d'observabilité
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
Disponibilité	Part des exécutions réussies (`OK` ou `WARNING`) parmi les exécutions évaluées sur une période ; les exécutions `UNKNOWN` en sont exclues. Voir chapitre 7 du guide d'utilisation et de supervision
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 8 du guide d'installation et d'administration
Dynatrace	Plateforme d'observabilité. Elle reçoit les données de la sonde via son API OTLP. Voir chapitre 2 du guide d'intégration et d'observabilité
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

Terme	Définition
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 3 du guide d'installation et d'administration
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 4 du guide de développement des scénarios
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 § 8.2
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 chapitre 2 du guide d'intégration et d'observabilité et chapitre 8 du guide d'installation et d'administration
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 7 du guide d'utilisation et de supervision
Fenêtre d'activité	Plage horaire pendant laquelle un plan d'exécution est actif (`activeWindows`), par exemple les heures ouvrées
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 8 du guide d'installation et d'administration
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 4 du guide d'installation et d'administration
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 3 du guide de développement des scénarios

Terme	Définition
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 § 8.6.4 du guide d'installation et d'administration, car une carte achetée ailleurs démarrerait en mode restreint. Voir chapitre 8 du guide d'installation et d'administration
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 9
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
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 8 du guide d'installation et d'administration
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 4 et 5 du guide de développement des scénarios
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

Terme	Définition
Log (journal)	Enregistrement horodaté d'un événement. Les logs d'exécution de la sonde peuvent être exportés via OpenTelemetry. Voir chapitre 2 du guide d'intégration et d'observabilité
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 9
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 § 8.3.4 du guide d'installation et d'administration
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 § 2.2
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 1 du guide d'utilisation et de supervision
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 5 du guide d'utilisation et de supervision
Observabilité	Capacité à comprendre l'état d'un système à partir des données qu'il émet : métriques, traces et logs
Onduleur	Alimentation secourue qui protège la sonde des coupures et micro-coupures électriques. Recommandé. Voir chapitre 8 du guide d'installation et d'administration
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 3
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

Terme	Définition
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 9
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 8 du guide d'installation et d'administration
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 9
Plan d'exécution (*playlist*)	Liste ordonnée de contrôles exécutés en séquence selon une planification commune. Voir chapitre 5
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 8 du guide d'installation et d'administration
Procédure stockée	Programme enregistré dans une base de données, exécutable par un contrôle `database` (champ `procedure`). Voir chapitre 5 du guide d'utilisation et de supervision
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 8 du guide d'installation et d'administration
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 2 du guide d'intégration et d'observabilité
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 9
Récepteur OTLP	Composant qui reçoit des données OTLP : collecteur OpenTelemetry, Agent Datadog, Prometheus lancé avec `--web.enable-otlp-receiver`. Voir chapitre 2 du guide d'intégration et d'observabilité
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 8 du guide d'installation et d'administration
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 1 du guide d'utilisation et de supervision
SaaS	Logiciel fourni sous forme de service hébergé par son éditeur (Datadog, Dynatrace, Grafana Cloud, par exemple)

Terme	Définition
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 chapitre 3 et chapitre 7 du guide d'utilisation et de supervision
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 1 du guide d'utilisation et de supervision
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 9
SOAP	Protocole d'échange de messages XML utilisé par certains web services. Il se supervise par un contrôle `http` en `POST` XML. Voir chapitre 5 du guide d'utilisation et de supervision
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 2, 3 et 8 du guide d'installation et d'administration
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 9
`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 4 et § 8.2
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 § 1.6.1 du guide d'utilisation et de supervision
`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`, § 3.1.3). Ces identifiants sont stables et indépendants du nom commercial SYNOVEX (chapitre 2)

Terme	Définition
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 § 8.1
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 4 du guide de développement des scénarios
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 9
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 2 du guide d'intégration et d'observabilité
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

Chapitre 11 — Licences tierces et mentions

11.1 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
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.

Chapitre 12 — Schémas JSON

Ce chapitre présente, pour chaque schéma JSON de référence (JSON Schema 2020-12), ses propriétés de premier niveau : nom, type, caractère obligatoire et description. Il est produit automatiquement à partir des fichiers de schémas : la structure présentée est donc toujours celle des schémas de référence. Les définitions imbriquées et les règles de validation détaillées figurent dans les fichiers eux-mêmes.

Les fichiers de schémas sont livrés dans l'archive `synovex-exemples.zip` (dossier `schemas/`), fournie avec les PDF de la bibliothèque documentaire et téléchargeable depuis la page d'accueil de la documentation en ligne. Utilisez-les pour valider vos fichiers de configuration et les réponses de l'API dans vos propres outils (validateur conforme à JSON Schema 2020-12).

12.1 Réponse de `GET /api/v1/checks/{id}/aggregates`

Fichier de référence : `aggregates.schema.json`.

Agrégats horaires d'un contrôle : ?from=&to=&step=1h. Par heure : nombre d'exécutions, nombre d'échecs, p50 et p95 de la durée. Rétention par défaut : 400 jours.

Propriété	Type	Obligatoire	Description
`checkId`	string	oui	
`step`	`1h`	oui	
`items`	tableau de object	oui	
`nextCursor`	string	non	

12.2 Certificat TLS de la sonde

Fichier de référence : `certificate.schema.json`.

Remplacement du certificat autosigné par un certificat de l'entreprise. Mode recommandé : `POST /api/v1/system/certificate/csr` (`$defs/csrRequest`) génère la clé sur la sonde, qui ne la quitte jamais, et renvoie la demande de certificat (CSR) au format PEM ; le certificat émis est ensuite importé par `PUT /api/v1/system/certificate` avec `$defs/import`, sans clé. Autre mode : import d'un PEM (certificat, chaîne, clé) ou d'un PKCS#12 protégé. Tant qu'aucun certificat importé n'est en place, le certificat autosigné est régénéré si le nom d'hôte, le nom mDNS ou l'adresse IPv4 change.

Définitions internes : `csrRequest`, `import`.

12.3 Réponse de `GET /api/v1/checks/{id}/status`

Fichier de référence : `check-status.schema.json`.

Dernier état consolidé d'un contrôle (rôle metrics). Utilisé par le plugin check_sva : STALE y produit le code 3.

Propriété	Type	Obligatoire	Description
`checkId`	string	oui	
`status`	valeurs : `OK` , `WARNING` , `CRITICAL` , `UNKNOWN` , `STALE`	oui	
`since`	string (format `date-time`)	oui	Début de l'état courant.
`lastRun`	object ou null	oui	Dernière exécution ; null si le contrôle n'a jamais été exécuté.

12.4 Contrôle

Fichier de référence : `check.schema.json` .

Propriété	Type	Obligatoire	Description
`id`	string	oui	
`name`	string	oui	
`description`	string	non	
`type`	valeurs : `icmp` , `tcp` , `dns` , `http` , `api` , `nagios` , `database` , `browser`	oui	
`enabled`	boolean	non	
`tags`	voir tags	non	
`schedule`	voir schedule ou null	non	Null : contrôle exécuté uniquement via un plan ou manuellement.
`timeout`	string	non	
`ipFamily`	voir ipFamily	non	Paramètre commun : famille d'adresses utilisée pour joindre la cible du contrôle. auto par défaut. ICMP en IPv6 par fping (ICMPv6).
`retry`	object	non	
`thresholds`	object	non	
`secrets`	tableau de string	non	Secrets que ce contrôle est autorisé à recevoir. Toute référence ${secret:X} doit y figurer.
`icmp`	object	non	
`tcp`	object	non	
`dns`	object	non	
`http`	voir httpRequest	non	
`api`	object	non	
`nagios`	object	non	
`database`	objet ou objet	non	
`browser`	object	non	

Définition interne	Type	Description
`browserStep`	object	
`extractRule`	object	
`httpRequest`	object	Requête HTTP. Les propriétés inconnues sont refusées : unevaluatedProperties: false est posé aux points d'utilisation (http, api.requests[]), ce qui équivaut à additionalProperties: false tout en permettant au contrôle api d'ajouter name et extract.
`proxyBypass`	`none`	Contourne le proxy global pour ce contrôle (http, api, browser). Absent : le proxy global et ses exclusions s'appliquent.

12.5 Définitions communes

Fichier de référence : `common.schema.json`.

Définitions partagées par les autres schémas, qui les référencent sous la forme common.schema.json#/$defs/. Ce fichier ne décrit aucune ressource de l'API à lui seul ; chaque définition est décrite dans le tableau ci-dessous.

Définition interne	Type	Description
`dateTime`	string (format `date-time`)	Date RFC 3339 en UTC.
`duration`	string	Durée : 500ms, 30s, 5m, 1h.
`errorType`	valeurs : `timeout`, `dns_error`, `connection_refused`, `connection_reset`, `tls_error`, `certificate_expired`, `http_status`, `assertion_failed`, `element_not_found`, `navigation_error`, `script_error`, `browser_crash`, `plugin_error`, `database_error`, `auth_failed`, `secret_missing`, `resource_exhausted`, `runner_crash`, `agent_restart`, `no_data`, `maintenance`, `unknown`	Type d'erreur d'une exécution en échec, issu d'une liste fermée (§ 8.1). no_data : requête sans résultat évaluable (contrôle database, onEmpty par défaut), état UNKNOWN. secret_missing : un secret déclaré par le contrôle est absent de la sonde, par exemple après l'import d'une configuration sans ses secrets ; l'exécution n'a pas lieu, état UNKNOWN. maintenance : exécution interrompue à l'expiration du délai de drainage (updates.drainTimeout) au début de l'activation d'une mise à jour, état UNKNOWN.
`hostnameConfirmation`	object	Corps de confirmation des actions irréversibles ou lourdes : réinitialisation usine, activation du SSD, verrouillage de SSH. La valeur doit être le nom d'hôte actuel de la sonde.
`id`	string	Identifiant lisible, stable, utilisé comme attribut de télémétrie.
`ipFamily`	valeurs : `auto`, `ipv4`, `ipv6`	Famille d'adresses utilisée pour joindre la cible. auto : IPv6 si l'IPv6 de la sonde est activé (settings network.ipv6.enabled) et si la cible possède une adresse AAAA, sinon IPv4. ipv4 ou ipv6 : famille imposée ; ipv6 échoue si l'IPv6 de la sonde est désactivé.
`operator`	valeurs : `eq`, `ne`, `lt`, `lte`, `gt`, `gte`, `contains`, `notContains`, `matches`, `exists`	Opérateur de comparaison des assertions (contrôles http, api, browser) et des règles (contrôle database).
`role`	valeurs : `admin`, `operator`, `viewer`, `metrics`	Rôle unique d'un utilisateur ou d'un jeton d'API (B3-securite § 1.1).

Définition interne	Type	Description
`schedule`	objet ou objet	Planification propre : interval (durée) ou cron (expression), l'un ou l'autre ; timezone, fuseau IANA d'interprétation de l'expression cron, UTC par défaut ; jitter, décalage aléatoire maximal appliqué à chaque échéance.
`secretName`	string	Nom d'un secret : majuscules, chiffres et tiret bas, commençant par une lettre, 64 caractères au plus (par exemple BOUTIQUE_PASSWORD).
`status`	valeurs : `OK`, `WARNING`, `CRITICAL`, `UNKNOWN`, `STALE`	État d'un contrôle ou d'une exécution. STALE qualifie un contrôle actif dont aucune exécution ne s'est terminée depuis plus de deux intervalles de planification ; il ne s'applique jamais à une exécution.
`tags`	object	Étiquettes libres. Seules celles déclarées exportables dans les paramètres deviennent des attributs de télémétrie.
`timeOfDay`	string	Heure locale au format HH:MM.
`weekday`	valeurs : `mon`, `tue`, `wed`, `thu`, `fri`, `sat`, `sun`	Jour de la semaine : mon, tue, wed, thu, fri, sat ou sun.

12.6 Archive d'export de configuration

Fichier de référence : `config-export.schema.json`.

Archive produite par `GET /api/v1/config/export` (sans secret ni compte) ou par `POST /api/v1/config/export` avec la phrase de passe dans l'en-tête X-Config-Passphrase, sans corps (comptes et secrets inclus, chiffrés). Contient contrôles, plans, scripts, destinations et paramètres, sans la section network. `POST /api/v1/config/import` reçoit l'archive telle quelle (application/json) et renvoie `$defs/importReport`. Paramètres de l'import : ?dryRun=true valide sans appliquer ; ?newIdentity=true conserve l'identité de la sonde cible au lieu de restaurer instanceId ; phrase de passe dans l'en-tête X-Config-Passphrase, obligatoire si l'archive contient la partie encrypted. À l'export comme à l'import, la phrase de passe se transmet dans cet en-tête, jamais dans l'adresse, qui peut être journalisée. `$defs/passphrase` décrit la valeur de l'en-tête.

Propriété	Type	Obligatoire	Description
`formatVersion`	`1`	oui	
`exportedAt`	string (format `date-time`)	oui	
`instanceId`	string	oui	Service.instance.id de la sonde d'origine, restauré par défaut à l'import pour assurer la continuité des séries de métriques.
`checks`	tableau de voir check.schema.json	oui	
`plans`	tableau de voir plan.schema.json	oui	
`scripts`	tableau de object	oui	
`exporters`	tableau de voir exporter.schema.json	oui	

Propriété	Type	Obligatoire	Description
`settings`	object	oui	Sections de paramètres exportées. network est exclue. license et le certificat de la sonde ne sont pas exportés : ils sont propres à l'appareil. dot1x n'est pas exportée : l'identité et l'identifiant client 802.1X sont propres à l'appareil. Dans ssh, locked est ignoré à l'import : un import ne verrouille ni ne déverrouille SSH.
`encrypted`	object	non	Présent uniquement si une phrase de passe a été fournie. Le texte clair est un document JSON contenant les comptes (empreintes Argon2id des mots de passe) et les secrets. Chaque secret y est conservé avec sa liste allowedHosts, restaurée à l'import ; un secret provenant d'une archive antérieure, sans liste, est importé mais reste inutilisable (jamais injecté) jusqu'à ce qu'un admin définisse sa liste, et importReport le signale.

Définition interne	Type	Description
`importReport`	object	Compte rendu d'import : objets créés, modifiés ou supprimés, et erreurs (fonction absente de l'édition, secret manquant…).
`objectRefs`	tableau de object	
`passphrase`	string	Valeur de l'en-tête X-Config-Passphrase (export par `POST /api/v1/config/export` et import) : phrase de passe dont est dérivée la clé de chiffrement (Argon2id).

12.7 Résultat du test d'une destination d'export

Fichier de référence : `exporter-test-result.schema.json`.

`POST /api/v1/exporters/{id}/test` renvoie 202 Accepted et un en-tête Location vers ce résultat.

Propriété	Type	Obligatoire	Description
`status`	valeurs : `OK`, `ERROR`	oui	OK : point de test accepté par la destination.
`httpStatus`	integer ou null	oui	Code HTTP renvoyé par la destination ; null si aucune réponse HTTP n'a été obtenue (DNS, connexion, TLS, délai).
`durationMs`	integer	oui	Latence de l'envoi.
`message`	string	oui	Diagnostic : message d'erreur renvoyé par la plateforme ou erreur réseau, secrets masqués.

12.8 Destination d'export

Fichier de référence : `exporter.schema.json`.

Propriété	Type	Obligatoire	Description
`id`	string	oui	
`kind`	valeurs : `otlp`, `prometheus`	oui	
`enabled`	boolean	non	

Propriété	Type	Obligatoire	Description
`preset`	valeurs : `otlp-generic`, `datadog-agent`, `datadog-agentless`, `dynatrace`, `grafana-cloud`, `prometheus-otlp`	non	Préréglage qui fixe les valeurs par défaut adaptées à la plateforme ; les champs explicites priment.

12.9 Réponse de `GET /health`

Fichier de référence : `health.schema.json`.

Test de vivacité, sans authentification. Code 200 et status OK si le moteur principal répond. Pendant le redémarrage lui-même, la sonde ne répond pas. La réponse ne contient aucune information sensible : ni version, ni édition, ni nom d'hôte.

Propriété	Type	Obligatoire	Description
`status`	valeurs : `OK`, `MAINTENANCE`	oui	OK (HTTP 200) : le moteur principal répond. MAINTENANCE (HTTP 503) : état de maintenance.

12.10 Plan d'exécution

Fichier de référence : `plan.schema.json`.

Propriété	Type	Obligatoire	Description
`id`	string	oui	
`name`	string	oui	
`enabled`	boolean	non	
`schedule`	voir schedule	oui	
`activeWindows`	tableau de object	non	Plages pendant lesquelles le plan s'exécute (par exemple heures ouvrées), dans le fuseau de schedule.timezone. Absent = toujours. Une heure from postérieure à l'heure to indique une plage qui franchit minuit. Aucun résultat STALE hors plage active.
`stopOnFailure`	boolean	non	
`maxDuration`	string	non	Durée maximale d'une exécution du plan. Une fois atteinte, le contrôle en cours se termine, puis le plan s'arrête ; les contrôles non exécutés ne produisent aucun résultat.
`items`	tableau de object	oui	

12.11 Résultat d'exécution

Fichier de référence : `run-result.schema.json`.

Propriété	Type	Obligatoire	Description
`id`	string	oui	ULID
`checkId`	string	oui	

Propriété	Type	Obligatoire	Description
`checkRevision`	integer	oui	
`planRunId`	string ou null	non	
`trigger`	valeurs : `schedule`, `plan`, `manual`, `retry`, `dryRun`	non	
`status`	valeurs : `OK`, `WARNING`, `CRITICAL`, `UNKNOWN`, `STALE`	oui	
`attempt`	integer	non	
`startedAt`	string (format `date-time`)	oui	Date et heure de début de l'exécution (RFC 3339, UTC).
`durationMs`	integer	oui	
`error`	object ou null	non	
`steps`	tableau de object	non	
`measurements`	object	non	Mesures propres au type : phases HTTP, RTT ICMP, perte, jours avant expiration du certificat, valeurs d'indicateurs métier, perfdata Nagios.
`output`	string	non	Sortie texte d'un plugin Nagios, secrets masqués.
`artifacts`	tableau de object	non	
`trace`	object	non	

12.12 Valeur d'un secret

Fichier de référence : `secret.schema.json`.

Corps de `PUT /api/v1/secrets/{name}` : création ou rotation, en écriture seule. La valeur n'est jamais renvoyée par l'API, y compris par `GET /api/v1/secrets/{name}.` Le nom du secret figure dans le chemin et suit common.schema.json#/`$defs/secretName`. `GET /api/v1/secrets/{name}` (rôle admin) renvoie les métadonnées du secret, décrites par `$defs/metadata`, avec un en-tête ETag à présenter dans If-Match lors d'une rotation (PUT) ou d'une suppression (DELETE). La création présente If-None-Match: * (réponse 412 si le secret existe déjà).

Propriété	Type	Obligatoire	Description
`value`	string	oui	
`allowedHosts`	tableau de string	oui	Hôtes autorisés à recevoir le secret : noms ou motifs d'hôtes, avec un port éventuel (par exemple api.example.com, *.example.com, db.example.internal:5432). La sonde n'injecte le secret que dans une requête, une connexion ou une page dont l'hôte figure dans cette liste ; en cas de redirection vers un autre hôte, les en-têtes porteurs du secret ne sont pas retransmis. Seul un admin peut fixer ou modifier cette liste ; un operator peut référencer le secret sans le lire ni modifier allowedHosts. Les références sont revérifiées à l'import d'une configuration. Obligatoire dans chaque PUT, rotation comprise : l'admin relit la liste, avec l'ETag, par `GET /api/v1/secrets/{name}` (métadonnées, sans la valeur) avant de la renvoyer. La règle vaut pour tout élément porteur du secret (en-tête, corps, paramètre d'adresse), y compris à la destination d'une redirection ; une entrée avec port limite l'injection à ce port ; le contrôle porte sur la destination effective au moment de l'injection.

Définition interne	Type	Description
`metadata`	object	Représentation renvoyée par `GET /api/v1/secrets/{name}` (rôle admin), accompagnée d'un en-tête ETag. Elle ne contient jamais la valeur du secret. Elle peut comporter d'autres métadonnées, notamment la date de rotation affichée dans la liste des secrets (manuel, § 6.8.4 du guide d'installation et d'administration) ; le nom de ce champ n'est pas fixé par ce schéma.

12.13 Paramètres : section dot1x

Fichier de référence : `settings-dot1x.schema.json`.

GET et `PUT /api/v1/settings/dot1x.` L'identifiant client (clé et certificat) ne figure pas dans cette section : il est créé par `POST /api/v1/system/dot1x/csr` (clé générée sur la sonde, recommandé) puis installé, ou importé en PKCS#12, par `PUT /api/v1/system/dot1x/credential` (écriture seule, `$defs/credential`). Avant tout accès au réseau, la section dot1x de provision.yaml remplit le même rôle. En cas d'échec d'authentification, la sonde réessaie, affiche l'état sur la console HDMI et signale l'échec dans l'état de santé network (`GET /api/v1/system`). Métrique sva.dot1x.certificate.expiry ; avertissement 30 jours avant l'expiration du certificat client. L'exception par adresse MAC (MAB) reste l'alternative.

Propriété	Type	Obligatoire	Description
`enabled`	boolean	non	
`identity`	string	non	Identité EAP présentée au serveur RADIUS.
`caRef`	string	non	Autorité de certification du serveur RADIUS : id d'une autorité de la section trust.

Définition interne	Type	Description
`credential`	object ou object	Corps de `PUT /api/v1/system/dot1x/credential` (écriture seule, jamais relu). Soit le certificat émis pour la dernière CSR (format pem, sans clé), soit un PKCS#12 protégé contenant certificat, chaîne et clé.
`csrRequest`	object	Corps facultatif de `POST /api/v1/system/dot1x/csr.` Le sujet de la CSR est dot1x.identity.
`csrResponse`	object	Réponse de `POST /api/v1/system/dot1x/csr` : la clé privée est générée sur la sonde et ne la quitte jamais.

12.14 Paramètres : section license (lecture seule)

Fichier de référence : `settings-license.schema.json`.

`GET /api/v1/settings/license.` Section en lecture seule : le certificat d'édition est inscrit sur la carte lors de sa préparation ; aucune clé n'est saisie, PUT est refusé (405). Pas de date d'expiration, sauf pour une carte de démonstration.

Propriété	Type	Obligatoire	Description
`edition`	voir edition	oui	
`serialNumber`	string	oui	Numéro de série du produit, imprimé sur l'encart ou l'étiquette.

Propriété	Type	Obligatoire	Description
`certificateStatus`	valeurs : `valid`, `missing`, `invalid`, `card_mismatch`, `demo_expired`	oui	État du certificat d'édition. Toute valeur autre que valid place la sonde en mode restreint.
`demo`	boolean	oui	Vrai pour une carte de démonstration.
`demoEndsAt`	string (format `date-time`)	non	Fin de la période d'évaluation : 30 jours après l'initialisation. Présent seulement pour une carte de démonstration.

12.15 Paramètres : section logging

Fichier de référence : `settings-logging.schema.json`.

GET et `PUT /api/v1/settings/logging.` Les journaux système sont volatils (en mémoire) par défaut, pour préserver la carte microSD.

Propriété	Type	Obligatoire	Description
`persistentUntil`	string (format `date-time`) ou null	non	Active la persistance des journaux jusqu'à cette date, pour un diagnostic. Une date située plus de 72 h après l'enregistrement est refusée (422). À l'échéance, la persistance se désactive automatiquement. null : journaux volatils.

12.16 Paramètres : section network

Fichier de référence : `settings-network.schema.json`.

GET et `PUT /api/v1/settings/network.` Un PUT enregistre la configuration sans l'appliquer ; `POST /api/v1/settings/network/apply` l'applique (`$defs/applyResponse`), puis `POST /api/v1/settings/network/confirm` la confirme (`$defs/confirmRequest`, réponse `$defs/confirmResponse`), dans les 120 s et depuis la nouvelle adresse. Sans confirmation, la configuration précédente est rétablie. Section exclue de l'export de configuration. IPv4 obligatoire ; IPv6 activable par l'objet ipv6, désactivé par défaut.

Propriété	Type	Obligatoire	Description
`hostname`	string	oui	Nom d'hôte de la sonde (étiquette DNS unique, en minuscules).
`mode`	valeurs : `dhcp`, `static`	oui	
`address`	string	non	Adresse IPv4 et longueur de préfixe (notation CIDR), en mode static.
`gateway`	string (format `ipv4`)	non	
`dns`	tableau de string (format `ipv4`) ou string (format `ipv6`)	non	Serveurs DNS, IPv4 ou IPv6. En mode dhcp, absents : ceux fournis par DHCP.
`searchDomains`	tableau de string (format `hostname`)	non	Domaines de recherche DNS. En mode dhcp, s'ajoutent à ceux fournis par DHCP (option 119).
`mdns`	boolean	non	Annonce mDNS du nom <nom d'hôte>.local.
`ipv6`	voir ipv6	non	

Définition interne	Type	Description
`applyResponse`	object	
`confirmRequest`	object	
`confirmResponse`	object	Réponse de `POST /api/v1/settings/network/confirm.` Pendant la fenêtre de confirmation, la sonde présente encore l'ancien certificat, déjà vérifié par l'administrateur. Lorsque le changement entraîne la régénération du certificat autosigné, la réponse contient le nouveau certificat, transmis par ce canal vérifié, qui en fait une source de confiance ; il est activé 30 s après la confirmation.
`ipv6`	object	Configuration IPv6 de la prise Ethernet. Désactivée par défaut. Lorsqu'elle est activée, le pare-feu applique les mêmes règles qu'en IPv4, les flux ICMPv6 de découverte des voisins sont autorisés, et le certificat autosigné inclut l'adresse IPv6 statique éventuelle.

12.17 Paramètres : section proxy

Fichier de référence : `settings-proxy.schema.json`.

GET et `PUT /api/v1/settings/proxy` (rôle admin). Proxy HTTP(S) sortant global, utilisant la méthode CONNECT, avec une authentification Basic facultative et une liste d'exclusions. Il s'applique aux exports vers les plateformes SaaS, au dépôt de mises à jour et aux contrôles http, api et browser ; chacun de ces contrôles peut le contourner par "proxy": "none", sans pouvoir en définir un autre. Les contrôles icmp, tcp, dns et database, ainsi que NTP et la résolution DNS de la sonde, ne passent pas par le proxy. Correspond à la section proxy du fichier provision.yaml (url, noProxy). Seuls les champs documentés par le manuel figurent ici.

Propriété	Type	Obligatoire	Description
`url`	string (format `uri`)	non	Adresse du proxy, par exemple http://proxy.example.internal:3128.
`noProxy`	tableau de string	non	Destinations joignables sans proxy : suffixes de domaine (par exemple .example.internal), adresses ou plages d'adresses en notation CIDR (par exemple 192.0.2.0/24). Y inscrire toutes les destinations internes joignables directement, dont le collecteur OpenTelemetry interne.
`passwordSecret`	string	non	Nom du secret de la sonde qui contient le mot de passe du compte technique du proxy (authentification Basic). La configuration ne contient jamais la valeur du mot de passe.

12.18 Paramètres : section retention

Fichier de référence : `settings-retention.schema.json`.

GET et `PUT /api/v1/settings/retention.` Durées de conservation locales des données.

Propriété	Type	Obligatoire	Description
`runs`	voir days	non	Exécutions détaillées : résultats complets, étapes, mesures, erreurs.
`hourlyAggregates`	voir days	non	Agrégats horaires par contrôle.
`artifacts`	object	non	Captures d'écran, journaux de console, extraits réseau. La première limite atteinte s'applique.

Propriété	Type	Obligatoire	Description
`audit`	voir days	non	Journal d'audit.

Définition interne	Type	Description
`days`	string	Durée en jours : 14d, 400d.
`size`	string	Taille : 500MB, 2GB.

12.19 Paramètres : section security

Fichier de référence : `settings-security.schema.json`.

GET et `PUT /api/v1/settings/security.` Le pare-feu local (nftables) est actif par défaut ; il autorise en entrée 443/TCP, 80/TCP (redirection), 5353/UDP (mDNS) et 22/TCP si SSH est activé. Avec l'IPv6 activé, les mêmes règles s'appliquent en IPv6 et ICMPv6 est autorisé pour la découverte des voisins.

Propriété	Type	Obligatoire	Description
`allowedNetworks`	tableau de string ou string	non	Réseaux autorisés à joindre les ports 443 et 22, en notation CIDR, IPv4 ou IPv6. Absente ou vide : aucune restriction par adresse source.
`httpRedirect`	boolean	non	Port 80 : redirection permanente vers HTTPS, sans contenu servi. false ferme le port 80.
`session`	object	non	
`lockout`	object	non	

12.20 Paramètres : section ssh

Fichier de référence : `settings-ssh.schema.json`.

GET et `PUT /api/v1/settings/ssh.` Accès de maintenance SSH temporaire, désactivé par défaut. Authentification par clé publique uniquement, sur un compte dédié non privilégié, avec élévation de droits journalisée. Seul un compte admin de la sonde peut l'activer ; l'éditeur ne dispose d'aucun accès permanent. Activation, connexions et désactivation sont inscrites au journal d'audit. À l'échéance enabledUntil, SSH se désactive et les sessions ouvertes sont fermées. Verrouillage définitif : `POST /api/v1/settings/ssh/lock` (corps common.schema.json#/`$defs/hostnameConfirmation`) ou provision.yaml (ssh.locked: true) ; irréversible jusqu'à une réinitialisation usine. Une sonde verrouillée refuse tout PUT qui active SSH.

Propriété	Type	Obligatoire	Description
`enabled`	boolean	non	
`enabledUntil`	string (format `date-time`)	non	Échéance de l'accès SSH. Obligatoire lorsque enabled vaut true. Durée proposée par défaut : 4 h ; au plus 72 h après l'activation : une échéance plus lointaine est refusée (422).
`authorizedKeys`	tableau de string	non	Clés publiques autorisées, au format OpenSSH (une ligne authorized_keys par élément).
`locked`	boolean	non	Verrouillage définitif de SSH. En lecture seule dans cette section : ne passe à true que par `POST /api/v1/settings/ssh/lock` ou par provision.yaml, et ne revient à false qu'après une réinitialisation usine.

12.21 Paramètres : section trust

Fichier de référence : `settings-trust.schema.json`.

GET et `PUT /api/v1/settings/trust.` Autorités de certification internes reconnues par la sonde. Ce magasin s'ajoute au magasin système et s'applique aux contrôles http, api, tcp (TLS), browser et aux destinations d'export. Une autorité peut aussi être désignée explicitement par son id (propriété tls.caRef d'un contrôle ou d'une destination).

Propriété	Type	Obligatoire	Description
`authorities`	tableau de object	non	

12.22 Paramètres : section updates

Fichier de référence : `settings-updates.schema.json`.

GET et `PUT /api/v1/settings/updates.` Le téléchargement et l'installation sur l'emplacement inactif ont lieu dès qu'une mise à jour est disponible, sans interrompre les contrôles ; seule l'activation (redémarrage sur le nouvel emplacement) interrompt la supervision ; les activations automatiques ordinaires et les correctifs critiques en next-window attendent une fenêtre du client ; les activations manuelles et les correctifs critiques en asap n'en dépendent pas ; préavis selon la matrice du § 8.4.2 du guide d'installation et d'administration ; toute activation est précédée d'un drainage (drainTimeout). Les correctifs de sécurité et les versions fonctionnelles (canal feature, réservées aux sondes dont le justificatif de maintenance est valide) ont des canaux et des images distincts, réglés séparément : un correctif de sécurité n'exige jamais d'installer une version fonctionnelle. Aucune fenêtre par défaut : sans fenêtre définie, aucune activation automatique n'a lieu ; la mise à jour reste en attente (état de santé updates en WARNING) et un administrateur l'active par `POST /api/v1/updates/apply`, immédiatement ou à une heure donnée (scheduledAt). Report : `POST /api/v1/updates/postpone.` Seuls échanges avec l'éditeur : vérification et téléchargement des mises à jour.

Propriété	Type	Obligatoire	Description
`security`	object	non	Canal security : correctifs de sécurité.
`feature`	object	non	Canal feature : versions fonctionnelles, canal et images distincts du canal security.
`windows`	tableau de voir window	non	Fenêtres d'activation du client, de 0 à 7. Elles conditionnent les activations automatiques ordinaires et les correctifs critiques en next-window ; elles ne conditionnent ni les activations manuelles ni les correctifs critiques en asap. Aucune fenêtre par défaut : liste vide ; une mise à jour ordinaire ou un correctif critique en next-window reste alors en attente jusqu'à son activation par un administrateur. L'assistant de mise en service impose de définir au moins une fenêtre ou de choisir le mode manuel ; dans provision.yaml, la section updates est facultative et ne définit aucune fenêtre par défaut. Aucune plage n'est imposée. Pour plusieurs sondes d'un même site, choisir des fenêtres qui ne se chevauchent pas.
`notice`	string	non	Préavis des activations automatiques ordinaires : l'activation a lieu au début de la première fenêtre qui commence au moins notice après l'annonce (bandeau de l'interface, `GET /api/v1/updates` scheduledActivation, journal d'audit, événement OTLP update.scheduled, métriques sva.update.pending, sva.maintenance.next_start et sva.maintenance.next_end). notice ne s'applique pas aux autres cas de la matrice du § 8.4.2 du guide d'installation et d'administration : activation manuelle immédiate, sans préavis ; activation manuelle programmée, préavis égal au délai jusqu'à scheduledAt (noticeReduced: true s'il est inférieur à notice) ; correctif critique, préavis de 15 minutes (next-window) ou de 15 minutes augmentées de criticalStagger (asap).

Propriété	Type	Obligatoire	Description
`drainTimeout`	string	non	Délai de drainage : au début de l'activation, la sonde passe en état de maintenance ; aucune exécution nouvelle ne démarre et celles en cours se terminent dans ce délai, au-delà elles sont interrompues en UNKNOWN avec le type d'erreur maintenance. 5m par défaut, 30m au maximum (au-delà, 422, contrôle hors schéma).
`repositoryUrl`	string (format `uri`)	non	Adresse du dépôt de mises à jour. Facultatif : à défaut, adresse du profil éditeur signé. Permet un miroir interne ; les images restent signées et vérifiées.

Définition interne	Type	Description
`mode`	valeurs : `automatic`, `manual`	
`window`	object	Fenêtre d'activation des mises à jour, en heure locale de la sonde. L'activation se termine par un redémarrage sur le nouvel emplacement système, suivi de l'autotest de 10 minutes.

12.23 Réponse de `GET /api/v1/system`

Fichier de référence : `system.schema.json`.

État détaillé de la sonde (rôle viewer) : version, édition, matériel, durée de fonctionnement, ressources, état des composants et des destinations d'export. Expose aussi la fin de la période de support et du justificatif de maintenance, la capacité mémoire (resources.memoryTotalBytes) et l'état du certificat d'édition (editionCertificateStatus), facultatifs dans le schéma.

Propriété	Type	Obligatoire	Description
`version`	string	oui	Version de l'image de la sonde.
`edition`	voir edition	oui	
`editionCertificateStatus`	valeurs : `valid`, `missing`, `invalid`, `card_mismatch`, `demo_expired`	non	État du certificat d'édition, mêmes valeurs que certificateStatus de `GET /api/v1/settings/license.` Toute valeur autre que valid indique le mode restreint et sa cause, lisibles avec le rôle viewer ; le détail figure dans la section license (rôle admin).
`hostname`	string	oui	
`site`	string	non	Site déclaré à la mise en service, s'il existe.
`uptimeSeconds`	integer	oui	
`supportEndsAt`	string (format `date-time`)	oui	Fin de la période de support de sécurité : au moins 5 ans après la mise sur le marché de la version majeure. Le contrat indique le mois et l'année ; la valeur est le dernier instant de ce mois, en UTC.
`maintenanceEndsAt`	string (format `date-time`) ou null	non	Fin de validité du justificatif de maintenance importé ; null s'il n'y en a pas. Son échéance n'affecte aucune fonction acquise.

Propriété	Type	Obligatoire	Description
`versions`	object	non	Versions des composants embarqués, dont puppeteer et chromium : c'est la version de Puppeteer exposée aux scripts.
`components`	object	oui	
`resources`	object	oui	
`exporters`	tableau de object	oui	

Définition interne	Type	Description
`componentStatus`	valeurs : `OK`, `WARNING`, `CRITICAL`, `UNKNOWN`	État d'un indicateur de santé de la sonde.
`edition`	valeurs : `essential`, `professional`, `enterprise`	

12.24 Jeton d'API

Fichier de référence : `token.schema.json`.

Jeton d'authentification à l'API. La sonde ne conserve qu'une empreinte du jeton ; sa valeur n'est renvoyée qu'une fois, dans la réponse de création. `POST /api/v1/tokens` (rôle admin) reçoit l'objet create et renvoie l'objet created ; `GET /api/v1/tokens` renvoie une liste d'objets read ; `DELETE /api/v1/tokens/{id}` révoque un jeton. Objet create : name (obligatoire, texte de 1 à 64 caractères qui indique l'usage, par exemple prometheus-prod), role (obligatoire : admin, operator, viewer ou metrics), expiresAt (facultatif, date RFC 3339 en UTC ; absent, le jeton n'expire pas). Objet read : id (identifiant attribué par la sonde, utilisé dans le chemin `/api/v1/tokens/{id}`), name, role, expiresAt (date, ou null si le jeton n'expire pas) et owner (identifiant de connexion de l'utilisateur qui a créé le jeton) ; la valeur du jeton n'y figure jamais. Objet created : propriétés de read, plus token, la valeur du jeton, affichée cette seule fois. Un jeton est immuable : aucune requête PUT ni PATCH ne le modifie. Pour changer son nom, son rôle ou sa date d'expiration, ou pour le renouveler, créez un nouveau jeton, mettez à jour l'outil qui l'utilise, puis révoquez l'ancien.

Définition interne	Type	Description
`create`	object	Corps de `POST /api/v1/tokens.`
`created`	object	Réponse de `POST /api/v1/tokens`, seule à contenir la valeur du jeton (token).
`name`	string	Nom descriptif indiquant l'usage, par exemple prometheus-prod.
`read`	object	Élément de la liste renvoyée par `GET /api/v1/tokens`, sans la valeur du jeton.

12.25 Réponse de `GET /api/v1/updates`

Fichier de référence : `updates-status.schema.json`.

État des mises à jour, rôle admin. La microSD porte deux emplacements système (A et B) en lecture seule ; une mise à jour, image signée (bundle), s'installe sur l'emplacement inactif, puis la sonde redémarre une seule fois sur celui-ci (tryboot). Si le moteur principal, l'API et l'autotest ne sont pas sains dans les 10 minutes, la sonde revient automatiquement à l'emplacement précédent et restaure les bases sauvegardées avant migration. `$defs/maintenanceResponse` décrit la réponse de `PUT /api/v1/system/maintenance.` Seule l'activation (redémarrage) attend une fenêtre du client ou une action d'un administrateur. state, scheduledActivation et

postponedUntil exposent l'activation à venir ; `$defs/applyRequest` et `$defs/postponeRequest` décrivent les corps de `POST /api/v1/updates/apply` et `POST /api/v1/updates/postpone.` `$defs/applyResponse` décrit la réponse de `POST /api/v1/updates/apply`, avec noticeReduced. Matrice unique des cas d'activation.

Propriété	Type	Obligatoire	Description
`slots`	tableau de voir slot	oui	Les deux emplacements système et la version de chacun.
`activeSlot`	voir slotName	oui	Emplacement sur lequel la sonde a démarré.
`available`	tableau de voir availableUpdate	oui	Mises à jour disponibles, d'après la dernière vérification en ligne ou le dernier bundle importé. Liste vide si aucune.
`state`	valeurs : `idle`, `installing`, `pending`, `scheduled`, `postponed`, `maintenance`	oui	Idle : aucune mise à jour en attente ; installing : téléchargement et installation sur l'emplacement inactif, sans interruption des contrôles ; pending : mise à jour installée, en attente d'activation (aucune fenêtre ou mode manual ; état de santé updates en WARNING, ou CRITICAL pour un correctif critique) ; scheduled : activation programmée (scheduledActivation.at) ; postponed : activation reportée (postponedUntil) ; maintenance : état de maintenance, drainage puis redémarrage et autotest.
`staged`	voir staged ou null	oui	Image préparée dans l'emplacement inactif (version, canal, caractère critique) ; null si aucune image n'est préparée.
`scheduledActivation`	voir scheduledActivation ou null	oui	Activation annoncée ; null si aucune activation n'est planifiée (aucune mise à jour en attente, ou mise à jour en attente sans instant déterminable).
`postponedUntil`	string (format `date-time`) ou null	oui	Fin du report demandé par `POST /api/v1/updates/postpone` ; null sans report en cours.
`lastCheckAt`	string (format `date-time`) ou null	non	Dernière vérification auprès du dépôt de l'éditeur ; null si la sonde n'a jamais vérifié (fonctionnement hors ligne).
`supportEndsAt`	string (format `date-time`)	oui	Fin de la période de support de sécurité, identique à `GET /api/v1/system.`
`maintenanceEndsAt`	string (format `date-time`) ou null	oui	Fin de validité du justificatif de maintenance ; null si aucun justificatif n'a été importé.
`lastResult`	object ou null	non	Résultat de la dernière tentative de mise à jour, retour arrière compris, quelle que soit l'activation concernée ; null si aucune mise à jour n'a été tentée. Un orchestrateur ne s'y fie que si activationId correspond à l'activation qu'il a lancée : un succès antérieur ne prouve pas la reprise.

Définition interne	Type	Description
`activationId`	string (format `uuid`)	Identifiant unique d'une activation planifiée. Conservé tant que la même image préparée reste à activer (report, nouvelle date, activation manuelle qui avance l'activation annoncée, recalcul après changement de fuseau) ; nouveau pour une nouvelle image. Repris par l'attribut sva.update.activation.id des événements update.scheduled, update.cancelled, maintenance.started et maintenance.ended.

Définition interne	Type	Description
`activationKind`	valeurs : `manual`, `manual-scheduled`, `automatic`, `critical-next-window`, `critical-asap`	Cas d'activation de la matrice du § 8.4.2 du guide d'installation et d'administration, identique à l'attribut sva.update.activation.kind : manual, activation manuelle immédiate ; manual-scheduled, activation manuelle programmée (scheduledAt) ; automatic, activation automatique ordinaire dans une fenêtre ; critical-next-window, correctif critique activé au début de la première fenêtre commençant au moins 15 minutes après l'annonce ; critical-asap, correctif critique activé 15 minutes après l'annonce, augmentées de security.criticalStagger.
`announcementSequence`	integer	Numéro de la dernière annonce de l'activation, identique à l'attribut sva.update.announcement.sequence : strictement croissant pour une même sonde, toutes annonces confondues (update.scheduled, update.cancelled) ; calculé comme le maximum entre le dernier numéro émis augmenté de 1 et l'horodatage UTC courant en millisecondes ; ne régresse ni après un redémarrage, ni après un retour arrière, ni après une restauration de configuration.
`applyRequest`	object	Corps facultatif de `POST /api/v1/updates/apply.` Sans scheduledAt : activation manuelle immédiate et sans préavis de la mise à jour installée sur l'emplacement inactif, avec drainage (drainTimeout), après confirmation explicite dans l'interface ; update.scheduled et maintenance.started sont émis aussitôt. Avec scheduledAt : activation programmée à cette heure, exposée dans scheduledActivation ; 422 si la date est passée ou située à moins d'une minute (contrôle hors schéma) ; acceptée avec noticeReduced: true si elle précède l'expiration du préavis notice. Une activation manuelle prime sur un report en cours. Réponse 202 Accepted (`$defs/applyResponse`) et Location vers le suivi. Journalisé (événement update.scheduled).
`applyResponse`	object	Réponse 202 Accepted de `POST /api/v1/updates/apply`, accompagnée de l'en-tête Location vers le suivi. activationId identifie l'activation à vérifier ensuite dans lastResult : si une activation était déjà annoncée pour la même image, son identifiant est conservé.
`availableUpdate`	object	
`maintenanceResponse`	object	Réponse de `PUT /api/v1/system/maintenance` après vérification de la signature du justificatif.
`postponeRequest`	object	Corps de `POST /api/v1/updates/postpone` : report de l'activation prévue jusqu'à until. Au plus 30 jours pour un correctif ordinaire et 7 jours pour un correctif critique ; au-delà, 422 (contrôle hors schéma). Journalisé dans l'audit.
`scheduledActivation`	object	Activation à venir de la mise à jour installée sur l'emplacement inactif.
`slot`	object	
`slotName`	valeurs : `A`, `B`	
`staged`	object	Image préparée dans l'emplacement inactif, en attente d'activation. Une seule image à la fois : une image security préempte une image feature déjà préparée, qui est abandonnée puis retéléchargée plus tard.

12.26 Compte de l'interface d'administration

Fichier de référence : `user.schema.json`.

Compte local de l'interface d'administration. Le mot de passe et son empreinte ne sont jamais renvoyés. `POST /api/v1/users` (rôle admin) reçoit l'objet create ; `GET /api/v1/users` renvoie une liste d'objets read ; `GET /api/v1/users/{id}` (rôle admin) renvoie un objet read, avec un en-tête ETag à présenter dans If-Match pour `PATCH`

`/api/v1/users/{id}` ; `PATCH /api/v1/users/{id}` (rôle admin) reçoit l'objet update ; `PUT /api/v1/users/me/password` (tout utilisateur authentifié) reçoit l'objet passwordChange ; `DELETE /api/v1/users/{id}` supprime le compte et révoque immédiatement ses sessions et ses jetons. Objet create : username (obligatoire, identifiant de connexion de 1 à 64 caractères, par exemple j.martin), role (obligatoire : admin, operator, viewer ou metrics) et password (obligatoire, mot de passe initial de 12 à 128 caractères, sans règle de composition ; son changement est imposé à la première connexion). Objet read : id (identifiant attribué par la sonde, utilisé dans le chemin `/api/v1/users/{id}`), username et role. Objet update : role et password, l'un des deux au moins ; l'identifiant de connexion ne se modifie pas. Objet passwordChange : currentPassword (mot de passe actuel de l'utilisateur authentifié) et newPassword (nouveau mot de passe, de 12 à 128 caractères), tous deux obligatoires. Effets sur les sessions et les jetons : un changement de rôle par PATCH ferme immédiatement les sessions de l'interface du compte ; un changement de mot de passe, par PATCH ou par `PUT /api/v1/users/me/password`, ferme les autres sessions de l'interface du compte et conserve 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.

Définition interne	Type	Description
`create`	object	Corps de `POST /api/v1/users.`
`passwordChange`	object	Corps de `PUT /api/v1/users/me/password` : changement de son propre mot de passe par tout utilisateur authentifié, qui fournit son mot de passe actuel. Exception à la règle de concurrence : ni If-Match ni If-None-Match n'est exigé, la preuve étant le mot de passe actuel. Mot de passe actuel erroné : réponse 403 avec le type d'erreur `/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.
`read`	object	Élément de la liste renvoyée par `GET /api/v1/users`, et représentation renvoyée par `GET /api/v1/users/{id}` (rôle admin) avec un en-tête ETag. Ni le mot de passe ni son empreinte n'y figurent.
`update`	object	Corps de `PATCH /api/v1/users/{id}` (rôle admin), présenté avec l'en-tête If-Match portant l'ETag renvoyé par `GET /api/v1/users/{id}` : modification du rôle, du mot de passe, ou des deux. L'identifiant de connexion (username) ne se modifie pas. Un changement de rôle ferme immédiatement les sessions de l'interface du compte ; les jetons d'API du compte conservent le rôle fixé à leur création.
`username`	string	Identifiant de connexion, par exemple j.martin.