Evidian

Guide de l'utilisateur de SafeKit

Logiciel de haute disponibilité pour applications critiques

Vue générale

Sujet	Ce document couvre toutes les phases de mise en œuvre de SafeKit : architecture, installation, tests, administration, résolution de problèmes, support, interface ligne de commande	
Public visé	***Architectures***	Aperçu technique
	Installation	Installation
	Console	La console web de SafeKit Sécurisation du service web de SafeKit
	Configuration avancée	Cluster.xml pour la configuration du cluster SafeKit Userconfig.xml pour la configuration du module Scripts du module pour la configuration du module Exemples de configurations de module
	Administration	Administration d'un module miroir Administration d'un module ferme Interface ligne de commande Administration avancée
	Support	Tests Résolution de problèmes Accès au support Evidian Index des messages du log
	Autres	Table des matières Logiciels tiers
Version	SafeKit 8.2	
OS supportés	Windows et Linux ; pour une liste détaillée des OS supportés, voir ici	
Sites web	Site marketing Evidian : http://www.evidian.com/safekit Site support Evidian : https://support.evidian.com/safekit	
Ref	39 F2 38MC 04	

Si vous avez des commentaires ou des questions relatives à ce document, contactez-nous via https://www.evidian.com/company/contact-evidian/

Table des matières

1. Aperçu technique

1.1 Généralités, solutions, architectures

1.1.1 Introduction à SafeKit

SafeKit, développé par Evidian, est une solution logicielle de haute disponibilité conçue pour garantir une disponibilité 24/7 des applications critiques pour les entreprises. Il prend en charge les plateformes Windows et Linux et élimine le besoin de disques partagés, d'éditions entreprises de bases de données ou de compétences techniques avancées, ce qui en fait une alternative rentable aux solutions de clustering traditionnelles.

Caractéristiques clés :

- Réplication synchrone en temps réel : Réplication continue des données entre les nœuds pour éviter toute perte de données.
- Basculement et retour automatiques : Basculement transparent vers un système secondaire en cas de panne et retour au système d'origine une fois opérationnel.
- Répartition de charge : Optimise l'utilisation des ressources en répartissant les charges de travail sur plusieurs serveurs.
- Indépendant de la plateforme : Compatible avec les machines physiques, les machines virtuelles et les infrastructures de cloud public.

Avantages clés :

- Aucune compétence spécifique : Aucune compétence informatique spécialisée requise pour le déploiement.
- Aucun surcoût matériel : Pas besoin de matériel spécifique comme des disques partagés ou des répartiteurs de charge.
- Aucun surcoût logiciel : Fonctionne avec les éditions standard de Windows et Linux.

Solutions clés :

- Niveau application : Haute disponibilité avec des scripts de redémarrage par application.
- Niveau hyperviseur : Haute disponibilité sans scripts de redémarrage par application.
- Niveau conteneur ou pod : Haute disponibilité sans scripts de redémarrage par application.

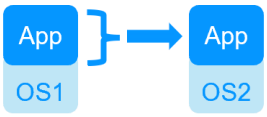
SafeKit est idéal pour les éditeurs de logiciels, les revendeurs et les distributeurs souhaitant améliorer leurs produits avec des fonctionnalités de haute disponibilité. Il offre également une opportunité OEM pour les partenaires d'intégrer SafeKit dans leurs propres applications.

1.1.2 Solutions SafeKit

Cliquez ici pour une liste des solutions SafeKit.

HA au niveau de l'application

Dans ce type de solution, seules les données applicatives sont répliquées. Et seule l'application est redémarrée en cas de panne.

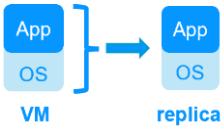


Des tâches d'intégration doivent être mises en œuvre : écrire les scripts de redémarrage pour l'application, définir les dossiers à répliquer, configurer les checkers logiciels, définir une adresse IP virtuelle.

Cette solution est indépendante de la plate-forme et fonctionne avec des applications à l'intérieur de machines physiques, de machines virtuelles et dans le cloud. Tous les hyperviseurs sont pris en charge (par exemple, VMware, Hyper-V, etc.).

HA au niveau de la machine virtuelle

Dans ce type de solution, l'intégralité de la machine virtuelle (VM) est répliquée, y compris l'application et le système d'exploitation. La machine virtuelle complète est redémarrée en cas de panne.



L'avantage est qu'il n'y a pas de scripts de redémarrage à écrire par application, ni d'adresse IP virtuelle à définir. Si vous ne savez pas comment fonctionne une application, c'est la solution la plus simple.

Cette solution fonctionne avec Windows/Hyper-V et Linux/KVM, mais pas avec VMware. Il s'agit d'une solution active/active avec plusieurs machines virtuelles répliquées et redémarrées entre les deux nœuds.

Note : Les applications exécutées dans des conteneurs ou des pods n'ont pas besoin non plus de scripts de redémarrage dédiés. SafeKit fournit des redémarrages génériques et une réplication en temps réel des données persistentes pour ces environnements (voir la liste des solutions SafeKit).

1.1.3 Architectures SafeKit

SafeKit propose deux clusters de haute disponibilité de base pour Windows et Linux :

- le cluster miroir, avec réplication de fichiers en temps réel et basculement, construit en déployant un module miroir sur 2 serveurs,
- Le cluster ferme, avec équilibrage de charge réseau et basculement, construit en déployant un module ferme sur 2 serveurs ou plus.

Plusieurs modules peuvent être déployés sur un même cluster. Ainsi, des architectures de clustering avancées peuvent être mises en œuvre :

- le cluster ferme+miroir construit en déployant un module ferme et un module miroir sur le même cluster,
- le cluster actif/actif construit en déployant plusieurs modules miroirs sur 2 serveurs,
- le cluster N-1 construit en déployant N modules miroirs sur N+1 serveurs.

Des clusters spécifiques sont également intéressants à prendre en compte avec SafeKit :

- le cluster Hyper-V ou KVM avec réplication en temps réel et basculement de machines virtuelles entières entre 2 hyperviseurs actifs,
- des clusters ferme ou miroir dans le Cloud.

1.1.4 Définition du cluster SafeKit

Un cluster SafeKit est un ensemble de serveurs sur lesquels SafeKit est installé et en cours d'exécution.

Tous les serveurs d'un cluster SafeKit donné partagent la même configuration de cluster, qui inclut la liste des serveurs et des réseaux utilisés. Ces serveurs communiquent entre eux pour maintenir une vue globale des configurations des modules SafeKit. Un serveur ne peut pas appartenir à plusieurs clusters SafeKit simultanément.

La configuration du cluster est une condition préalable à l'installation et à la configuration des modules SafeKit. Cela peut être fait à l'aide de la console Web de SafeKit ou de commandes en ligne.

1.1.5 Définition du module SafeKit

Un module est une personnalisation de SafeKit pour une application ou un hyperviseur spécifique. Cliquez ici pour une liste des modules et leurs guides d'installation rapide.

Types de modules

- Modules génériques ferme et miroir pour de nouvelles applications,
- Modules d'application préconfigurés pour des bases de données, des serveurs web...,
- Modules hyperviseurs (hyperv.safe, kvm.safe) pour la réplication en temps réel et le redémarrage de machines virtuelles complètes.

Contenu du module

En pratique, un module est un fichier « .safe » (type zip) qui comprend :

- Le fichier de configuration userconfig.xml, qui contient :
  - L'adresse IP virtuelle (non nécessaire pour un module hyperviseur),
  - Répertoires de fichiers à répliquer en temps réel (pour un module miroir),
  - Critères d'équilibrage de charge réseau (pour un module ferme),
  - Configuration des détecteurs de pannes logicielles et matérielles,
- Les scripts permettant de démarrer et d'arrêter une application ou une machine virtuelle.

Étapes de déploiement

Une fois qu'un module est configuré et testé, le déploiement ne nécessite aucune compétence informatique spécifique :

- Installer l'application ou l'hyperviseur sur 2 serveurs standards,
- Installez le logiciel SafeKit sur les deux serveurs,
- Installez le module sur les deux serveurs.

La configuration, le déploiement et la surveillance des modules peuvent être effectués à l'aide de la console Web de SafeKit ou de commandes en ligne.

1.1.6 Limites de SafeKit

Utilisation typique avec SafeKit			
Réplication de quelques téraoctets	Réplication < 1 millions de fichiers	Réplication <= 32 machines virtuelles	LAN 1 ou 10 G/s ou LAN étendu
Limitation			
La resynchronisation après une défaillance prend trop de temps. Sur un réseau de 1 Gbit/s, 3 heures pour 1 téraoctets. Sur un réseau 10 Gbit/s, 1 heure ou moins pour 1 téraoctets (dépend des performances d'E/S du disque en écriture).	La resynchronisation après une défaillance prend trop de temps. Temps de vérifier chaque fichier entre les deux nœuds.	En mode réplication complète de machines virtuelles, et avec une machine virtuelle dans un module miroir, la limite est de 32 modules par cluster.	Le basculement de l'adresse IP virtuelle est intégré lorsqu'on est dans le même sous-réseau. Un LAN fournit une bande passante adéquate pour la resynchronisation. Un LAN offre une latence adéquate (généralement un aller-retour de moins de 2 ms) pour la réplication synchrone.
Alternative			
Utilisez un stockage partagé.	Mettez les fichiers dans un disque dur virtuel répliqué par SafeKit.	Utilisez une autre solution de HA avec stockage partagé.	Utilisez des solutions de backup avec réplication asynchrone.

1.2 Le cluster miroir de SafeKit

1.2.1 Réplication de fichiers en temps réel et basculement d'application

Le cluster miroir est une solution de haute disponibilité active-passive, créée en déployant un module miroir au sein d'un cluster à deux nœuds. L'application s'exécute sur un serveur primaire et est redémarrée automatiquement sur un serveur secondaire en cas de défaillance du serveur principal.

Avec sa fonction de réplication de fichiers temps réel, cette architecture est particulièrement adaptée pour fournir une haute disponibilité aux applications back-end avec des données critiques à protéger contre les pannes.

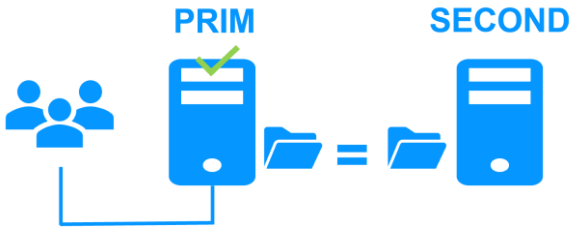
Les solutions Microsoft SQL Server, PostgreSQL, MariaDB, Oracle, Milestone, Nedap, Docker, Podman, Hyper-V et KVM sont des exemples de modules miroirs. Vous pouvez

créer votre propre module miroir pour votre application sur la base du module générique mirror.safe. Voir ici une liste des modules.

Notez que les modules miroirs Hyper-V et KVM répliquent des machines virtuelles entières, y compris les applications et les systèmes d'exploitation. Ils ne nécessitent pas d'adresse IP virtuelle, car le redémarrage de la machine virtuelle gère le basculement de l'adresse IP physique de la VM.

Le cluster miroir fonctionne comme suit.

1.2.2 Étape 1. Fonctionnement normal



Le serveur 1 (PRIM) exécute l'application.

SafeKit réplique les fichiers ouverts par l'application. Seules les modifications apportées par l'application dans les fichiers sont répliquées en temps réel sur le réseau, limitant ainsi le trafic.

Pour la réplication, seuls les noms des répertoires de fichiers à répliquer sont configurés dans SafeKit. Il n'y a pas de conditions préalables à l'organisation des disques pour les deux serveurs. Les répertoires à répliquer peuvent se trouver sur le disque système.

1.2.3 Étape 2. Basculement

Lorsque le serveur 1 tombe en panne, le serveur 2 prend le relais. SafeKit bascule l'adresse IP virtuelle et redémarre automatiquement l'application sur le serveur 2. L'application retrouve les fichiers répliqués par SafeKit à jour sur le serveur 2, grâce à la réplication synchrone entre le serveur 1 et le serveur 2. L'application continue de s'exécuter sur le serveur 2 en modifiant localement ses fichiers qui ne sont plus répliqués sur le serveur 1.

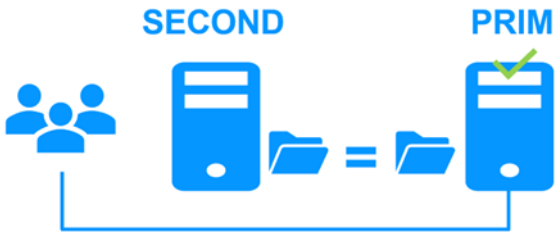
Le temps de basculement est égal au temps de détection des pannes (défini à 30 secondes par défaut) plus le temps de démarrage de l'application. Contrairement aux solutions de réplication de disque, il n'y a pas de délai pour le remontage des systèmes de fichiers et l'exécution des procédures de récupération.

1.2.4 Étape 3. Réintégration et resynchronisation

La réintégration consiste à redémarrer le serveur 1 après avoir résolu le problème qui l'a mis en panne. SafeKit resynchronise automatiquement les fichiers, en ne mettant à jour que les fichiers modifiés sur le serveur 2 lorsque le serveur 1 a été arrêté.

Cette réintégration automatique s'effectue sans arrêter l'application, qui peut continuer à s'exécuter sur le serveur 2. Il s'agit d'une caractéristique majeure qui différencie SafeKit des autres solutions, qui nécessitent des opérations manuelles pour réintégrer le serveur 1 dans le cluster.

1.2.5 Étape 4. Retour au fonctionnement normal



Après la réintégration, les fichiers sont à nouveau en mode miroir, comme à l'étape 1. Le système est de nouveau en mode haute disponibilité, l'application s'exécutant sur le serveur 2 et SafeKit répliquant les mises à jour des fichiers sur le serveur 1.

Si les administrateurs souhaitent que l'application s'exécute sur le serveur 1, ils peuvent exécuter une commande « Stop / Start » sur le serveur PRIM soit à travers la console au moment opportun, soit automatiquement via la configuration d'un serveur primaire par défaut.

1.2.6 Réplication synchrone et réplication asynchrone

Il existe une différence significative entre la réplication synchrone, telle qu'elle est proposée par la solution miroir de SafeKit, et la réplication asynchrone traditionnellement proposée par d'autres solutions de réplication de fichiers.

Avec la réplication synchrone, lorsqu'une E/S disque est effectuée par l'application sur le serveur primaire à l'intérieur d'un fichier répliqué, SafeKit attend l'accusé de réception d'E/S du disque local et du serveur secondaire, avant d'envoyer l'accusé de réception d'E/S à l'application. Ce mécanisme est essentiel pour la récupération des applications transactionnelles.

La latence d'un LAN (généralement un aller-retour de moins de 2 ms) entre les serveurs est nécessaire pour mettre en œuvre la réplication synchrone des données, éventuellement avec un LAN étendu dans deux salles informatiques géographiquement éloignées.

Avec la réplication asynchrone implémentée par d'autres solutions, les E/S sont placées dans un journal sur le serveur primaire, mais le serveur primaire n'attend pas les accusés de réception d'E/S du serveur secondaire. Ainsi, toutes les données qui n'ont pas été

copiées à travers le réseau sur le deuxième serveur sont perdues en cas de défaillance du premier serveur.

En particulier, une application transactionnelle peut perdre des données validées en cas de défaillance. La réplication asynchrone peut être utilisée pour la réplication de données sur un WAN à faible vitesse afin de sauvegarder des données à distance (backup), mais elle n'est pas adaptée à la haute disponibilité avec basculement automatique.

SafeKit fournit une solution semi-synchrone, mettant en œuvre l'asynchronisme non pas sur le serveur primaire mais sur le secondaire. Dans cette solution, SafeKit attend toujours l'accusé de réception des deux serveurs avant d'envoyer l'accusé de réception à l'application. Mais sur le secondaire, il y a 2 options asynchrone ou synchrone. Dans le cas asynchrone, le secondaire envoie l'accusé de réception au primaire à la réception de l'E/S et écrit sur le disque par la suite. Dans le cas synchrone, le secondaire écrit l'E/S sur le disque, puis envoie l'accusé de réception au primaire. Le mode synchrone est nécessaire si l'on considère une double panne de courant simultanée de deux serveurs, avec l'impossibilité de redémarrer l'ancien serveur primaire et la nécessité de redémarrer sur le serveur secondaire.

1.2.7 Comportement en cas d'isolation réseau

Un **heartbeat** est un mécanisme permettant de synchroniser deux serveurs et de détecter les défaillances en échangeant des données sur un réseau partagé. Si un serveur perd tous les heartbeats, il suppose que l'autre est en panne et exécute l'application seul (état ALONE).

SafeKit supporte plusieurs heartbeats sur plusieurs réseaux partagés. Un réseau dédié avec un deuxième heartbeat peut éviter l'isolation réseau et être également utilisé comme réseau de réplication.

Isolation réseau :

- Sur perte de tous les heartbeats, les deux serveurs passent à l'état ALONE, exécutant l'application indépendamment.
- Après l'isolation, un serveur s'arrête et resynchronise les données à partir de l'autre serveur.
- Le cluster revient à l'état PRIM-SECOND.

Checker splitbrain :

- Utilise une adresse IP témoin (généralement un routeur) pour éviter la double exécution pendant l'isolation réseau.
- Seul le serveur avec un accès au témoin passe ALONE ; l'autre attend (WAIT).
- Après l'isolation, le serveur WAIT se resynchronise et devient SECOND.

1.2.8 Réplication à 3 nœuds

SafeKit ne supporte la réplication qu'entre deux nœuds. Cependant, il est possible de mettre en œuvre une réplication à 3 nœuds en combinant SafeKit avec une solution de sauvegarde.

Une application est rendue hautement disponible entre 2 nœuds grâce à SafeKit avec sa réplication synchrone en temps réel (sans perte de données) et son basculement automatique. De plus, une solution de sauvegarde est mise en œuvre pour la réplication asynchrone vers un troisième nœud dans un site de reprise après sinistre. Étant donné qu'il y a une perte de données avec une solution de sauvegarde asynchrone, le basculement vers le troisième nœud est manuel et décidé par un administrateur.

Notez que la réplication en temps réel de SafeKit n'élimine pas la nécessité d'une solution de sauvegarde. Par exemple, une attaque par ransomware chiffrant les données répliquées sur le serveur primaire chiffrera également les données sur le serveur secondaire en temps réel avec SafeKit. Seule une solution de sauvegarde avec une politique de rétention peut résoudre une attaque par ransomware. L'administrateur doit restaurer la sauvegarde d'avant l'attaque par ransomware.

1.2.9 SafeKit sur un seul nœud pour résister aux pannes logicielles

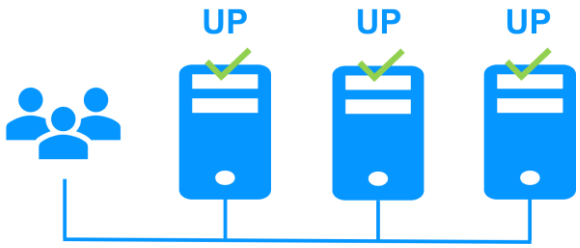
Vous pouvez configurer un module en mode "light", ce qui correspond à un module fonctionnant sur un seul nœud sans se synchroniser avec d'autres nœuds (contrairement aux modules miroir ou ferme). Un module "light" inclut le démarrage et l'arrêt d'une application, ainsi que les vérificateurs SafeKit qui détectent les erreurs logicielles et effectuent des redémarrages automatiques sur un seul nœud.

Le module "light" s'interface avec la console SafeKit, permettant à un administrateur de visualiser l'état du module applicatif et de déclencher manuellement des redémarrages de l'application via une interface clic-bouton.

Il n'est pas nécessaire de définir une adresse IP virtuelle ou des répertoires à répliquer dans un module "light". Notez que cela peut servir également de première étape avant de passer à un module miroir ou à un module ferme

1.3 Le cluster ferme de SafeKit

1.3.1 Équilibrage de charge réseau et basculement d'application



Le cluster ferme est une solution de haute disponibilité active-active, créée en déployant un module ferme au sein d'un cluster de deux nœuds ou plus. Le cluster ferme fournit à la fois l'équilibrage de la charge réseau, grâce à une distribution transparente du trafic réseau, et le basculement logiciel et matériel. Cette architecture offre une solution simple pour supporter l'augmentation de la charge du système.

La même application s'exécute sur chaque serveur, et la charge est équilibrée par la répartition de l'activité réseau sur les différents serveurs de la ferme.

Les clusters fermes sont adaptés aux applications frontales telles que les services Web.

Les solutions Apache, Microsoft IIS, NGINX sont des exemples de modules ferme. Vous pouvez écrire votre propre module ferme pour votre application, basé sur le module générique farm.safe. Voir ici pour une liste des modules.

1.3.2 Principe d'une adresse IP virtuelle avec équilibrage de charge réseau

L'adresse IP virtuelle est configurée localement sur chaque serveur de la ferme. Le trafic en entrée pour cette adresse est réparti entre tous les serveurs par un filtre au sein du noyau de chaque serveur.

L'algorithme d'équilibrage de charge à l'intérieur du filtre est basé sur l'identité des paquets clients (adresse IP du client, port TCP du client). En fonction de l'identité du

paquet client, un seul filtre sur un serveur accepte le paquet. Une fois qu'un paquet est accepté par le filtre sur un serveur, seuls le processeur et la mémoire de ce serveur sont utilisés par l'application répondant à la demande du client. Les messages de sortie sont envoyés directement du serveur d'application au client.

En cas de défaillance d'un serveur, le protocole de heartbeat de SafeKit dans une ferme reconfigure les filtres pour rééquilibrer le trafic entre les serveurs disponibles restants.

1.3.3 Équilibrage de charge pour les services Web avec ou sans état

Avec un serveur à état, l'affinité de session est requise. Le même client doit se connecter au même serveur sur plusieurs sessions TCP pour récupérer son contexte. Dans ce scénario, la règle d'équilibrage de charge dans SafeKit est configurée sur l'adresse IP du client. Cela garantit que le même client se connecte toujours au même serveur sur plusieurs sessions TCP, tandis que différents clients sont répartis sur différents serveurs de la ferme. Cette configuration est nécessaire lorsqu'il y a affinité de session.

Avec un serveur sans état, il n'y a pas d'affinité de session. Le même client peut se connecter à différents serveurs de la ferme sur plusieurs sessions TCP, car aucun contexte n'est stocké localement sur un serveur d'une session à l'autre. Dans ce cas, la règle d'équilibrage de charge dans SafeKit est configurée sur l'identité de session du client TCP. Cette configuration est optimale pour la distribution des sessions entre les serveurs, mais nécessite un service TCP sans affinité de session.

1.3.4 Solution de haute disponibilité en chaîne dans une ferme

Qu'est-ce qu'une solution de haute disponibilité en chaîne (également connue sous le nom de solution de haute disponibilité en cascade) ?

- Plusieurs serveurs sont liés en séquence : Si un serveur tombe en panne, le suivant dans la chaîne prend le relais.
- Gestion basée sur la priorité : Un seul serveur gère toutes les requêtes des clients, celui ayant la plus haute priorité dans la chaîne et qui est disponible.
- Processus de basculement : Si le serveur ayant la plus haute priorité tombe en panne, le prochain serveur disponible avec la plus haute priorité prend le relais.
- Réintégration : Lorsqu'un serveur revient en ligne et qu'il a la plus haute priorité, il reprend la gestion de toutes les requêtes des clients.
- Temps de récupération rapide : Cette solution a un temps de récupération rapide, car l'application est pré-démarrée sur tous les serveurs. Le temps de récupération est essentiellement le temps nécessaire pour reconfigurer les priorités entre les serveurs de la ferme (quelques secondes).
- Limitations de la réplication : Cette solution ne prend pas en charge la réplication en temps réel, qui est limitée à l'architecture miroir. Cependant, une architecture combinée ferme+miroir est disponible.

Pour mettre en œuvre une solution de haute disponibilité en chaîne, SafeKit offre une variable "power" dans les règles d'équilibrage de charge : elle se définit au niveau de chaque serveur du cluster. La variable power vous permet d'allouer plus ou moins de trafic à un serveur. Lorsque la variable power est définie comme un multiple de 64 entre les serveurs (par exemple, 1, 64, 64*64, 64*64*64, ...), la solution de haute disponibilité en chaîne est mise en œuvre

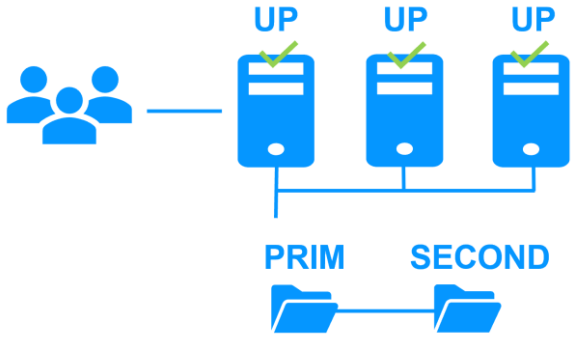
1.4 Clusters exécutant plusieurs modules

1.4.1 Le cluster ferme+miroir de SafeKit

Équilibrage de charge réseau, réplication de fichiers et basculement d'applications

Vous pouvez mélanger des modules ferme et miroir sur le même cluster.

Cette option vous permet d'implémenter une architecture d'application multiniveau, telle que apache_farm.safe (architecture ferme avec équilibrage de charge et basculement) et postgresql.safe (architecture miroir avec réplication de fichiers et basculement) sur les mêmes serveurs.

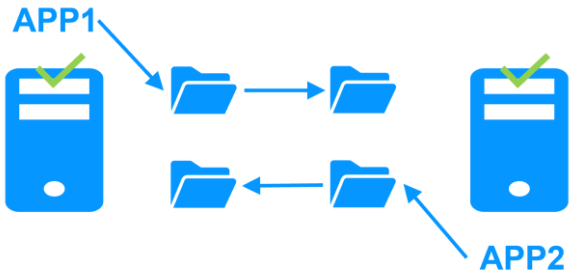


Par conséquent, l'équilibrage de charge, la réplication de fichiers et le basculement sont gérés de manière cohérente sur les mêmes serveurs.

1.4.2 Le cluster actif/actif avec réplication de SafeKit

Réplication croisée et basculement mutuel

Dans un cluster actif/actif avec réplication, il y a deux serveurs et deux modules miroirs en basculement mutuel (appli1.safe et appli2.safe). Chaque serveur d'application est une sauvegarde de l'autre serveur.



En cas de défaillance d'un serveur d'application, les deux applications s'exécutent sur le même serveur physique. Une fois le serveur défaillant redémarré, son application revient à son serveur primaire par défaut.

Un cluster de basculement mutuel est plus rentable que deux clusters miroirs distincts, car il élimine le besoin de serveurs de sauvegarde qui restent inactifs la plupart du temps, en attendant qu'un serveur principal tombe en panne. Toutefois, en cas de défaillance d'un serveur, le serveur restant doit être capable de gérer la charge de travail combinée des deux applications.

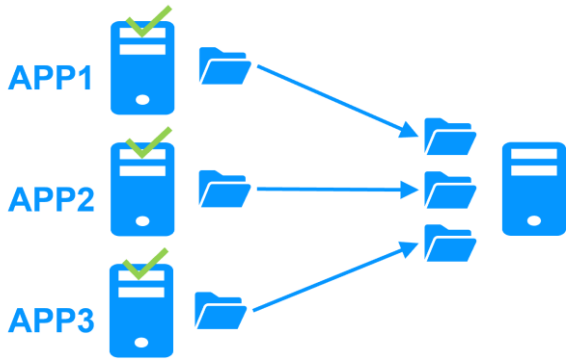
Notez que :

- Les deux applications, Appli1 et Appli2, doivent être installées sur chaque serveur pour permettre le basculement des applications.
- Cette architecture n'est pas limitée à deux applications ; N modules applicatifs peuvent être déployés sur les deux serveurs.
- Chaque module miroir aura sa propre adresse IP virtuelle, ses propres répertoires de fichiers répliqués et ses propres scripts de reprise.

1.4.3 Le cluster N-1 de SafeKit

Réplication et basculement d'applications de N serveurs vers 1

Dans un cluster N-1, N modules applicatifs miroirs sont déployés sur N serveurs principaux et un seul serveur de sauvegarde.



En cas de panne, contrairement à un cluster actif/actif, le serveur de sauvegarde n'a pas besoin de gérer une double charge de travail en cas de défaillance d'un serveur principal. Cela suppose qu'une seule défaillance se produit à la fois. Bien que la solution puisse prendre en charge plusieurs pannes de serveur principal simultanément, dans ce cas, le serveur de sauvegarde unique devra gérer la charge de travail combinée de tous les serveurs défaillants. Dans un cluster N-1, N modules applicatifs miroirs sont installés entre N serveurs principaux et un serveur de sauvegarde.

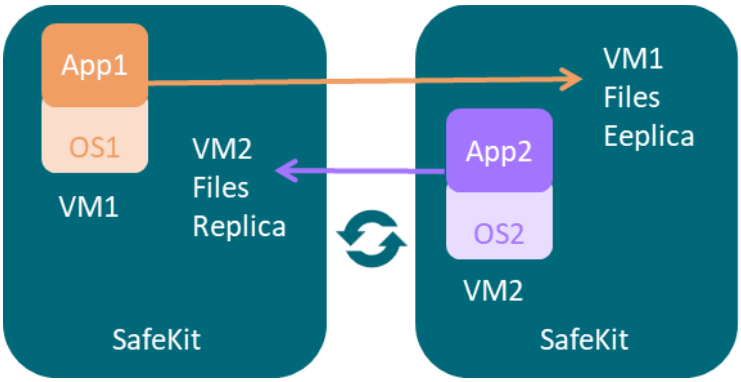
Notez que :

- Toutes les applications (Appli1, Appli2, Appli3) doivent être installées sur le serveur de sauvegarde unique pour permettre le basculement des applications.
- Chaque module miroir aura sa propre adresse IP virtuelle, ses propres répertoires de fichiers répliqués et ses propres scripts de reprise.

1.5 Le cluster Hyper-V ou KVM de SafeKit

1.5.1 Équilibrage de charge, réplication, basculement de machines virtuelles complètes

Le cluster Hyper-V ou KVM est un exemple de cluster actif-actif. Plusieurs applications peuvent être hébergées dans diverses machines virtuelles, qui sont répliquées et redémarrées par SafeKit. Chaque machine virtuelle est gérée par SafeKit au sein de son propre module miroir.



La solution présente les caractéristiques suivantes :

- Réplication synchrone en temps réel de machines virtuelles entières avec des capacités de basculement.
- Une console SafeKit centralisée et conviviale pour la gestion de toutes les machines virtuelles, y compris la possibilité de migrer des machines virtuelles entre les serveurs afin d'optimiser la répartition de la charge.
- Un checker pour chaque machine virtuelle afin de détecter si elle s'est verrouillée, s'est bloquée ou a cessé de fonctionner, et de redémarrer la machine virtuelle si nécessaire.
- Une solution attrayante qui ne nécessite aucune intégration d'application.
- Une architecture robuste adaptée aux solutions à haute disponibilité qui ne peuvent pas être intégrées au niveau applicatif.

Une version d'essai gratuite du cluster Hyper-V avec SafeKit est disponible ici.

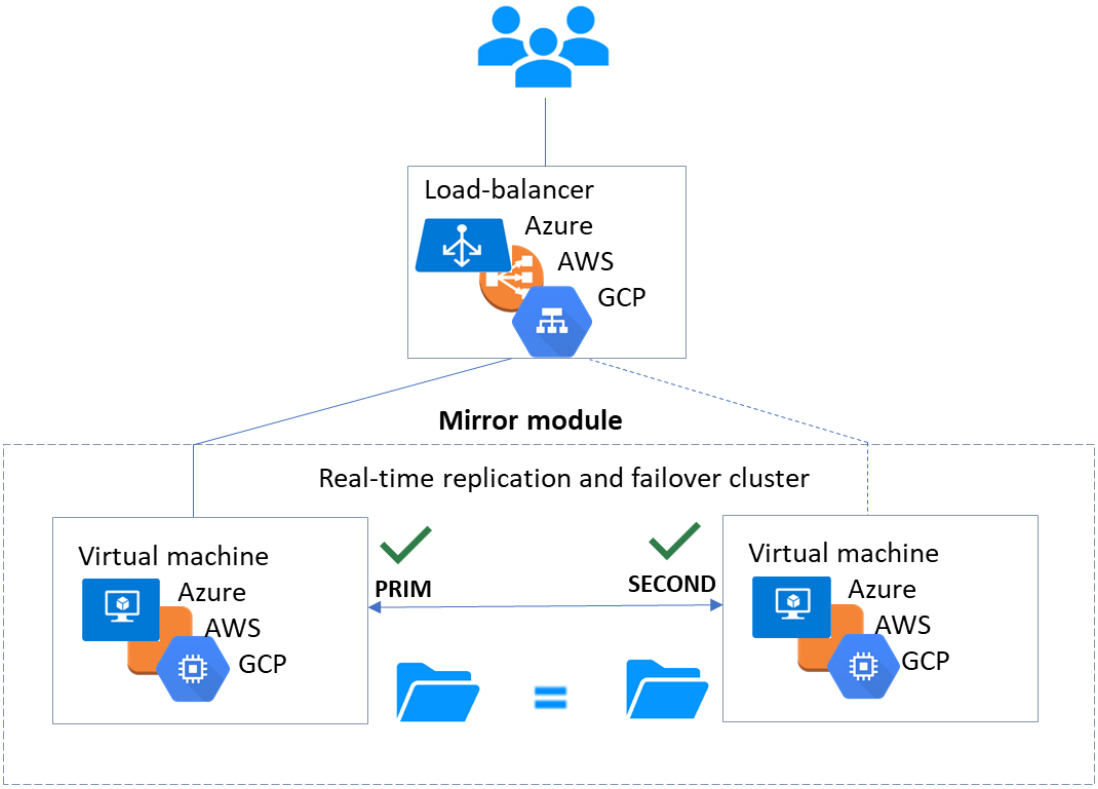
Une version d'essai gratuite du cluster KVM avec SafeKit est disponible ici.

1.6 Clusters SafeKit dans le cloud

Pour une description complète, voir la section 16.

1.6.1 Cluster miroir dans Azure, AWS et GCP

SafeKit fournit des clusters de haute disponibilité avec réplication en temps réel et basculement dans Azure, AWS et GCP grâce au déploiement d'un module miroir.



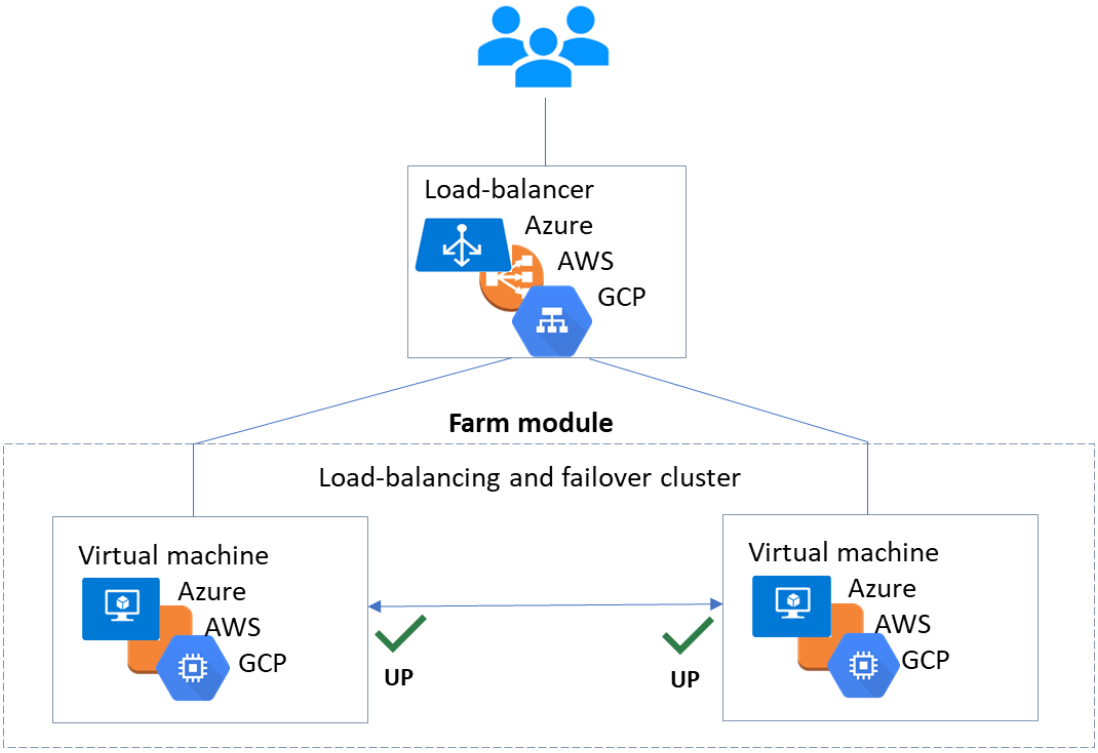
La solution miroir dans le cloud est similaire à celle sur site, sauf que l'adresse IP virtuelle doit être configurée au niveau de l'équilibreur de charge :

- Les machines virtuelles sont placées dans différentes zones de disponibilité, qui se trouvent dans des sous-réseaux différents.
- L'application critique s'exécute sur le serveur primaire.
- Les utilisateurs se connectent à une adresse IP virtuelle primaire/secondaire gérée par l'équilibreur de charge du cloud.
- SafeKit fournit un « health check » configuré dans l'équilibreur de charge. Sur le serveur primaire, le « health check » renvoie OK à l'équilibreur de charge, tandis qu'il ne renvoie rien sur le serveur secondaire. Ainsi, toutes les requêtes adressées à l'adresse IP virtuelle sont acheminées vers le serveur primaire.
- Si le serveur primaire tombe en panne ou est arrêté, le serveur secondaire devient automatiquement le serveur primaire et renvoie OK au « health check ». Ainsi, toutes les requêtes adressées à l'adresse IP virtuelle sont redirigées vers le nouveau serveur primaire.
- SafeKit surveille l'application critique sur le serveur primaire à l'aide des checkers de SafeKit.
- SafeKit redémarre automatiquement l'application critique en cas de défaillance logicielle ou matérielle, grâce à des scripts de redémarrage.
- SafeKit effectue une réplication synchrone en temps réel des fichiers contenant les données critiques.

Pour plus d'informations, consultez le cluster miroir dans Azure, le cluster miroir dans AWS ou le cluster miroir dans GCP.

1.6.2 Cluster ferme dans Azure, AWS et GCP

SafeKit fournit des clusters à haute disponibilité avec équilibrage de charge réseau et basculement dans Azure, AWS et GCP grâce au déploiement d'un module ferme.



La solution ferme dans le cloud est similaire à celle sur site, sauf que l'adresse IP virtuelle doit être configurée au niveau de l'équilibreur de charge :

- Les machines virtuelles sont placées dans différentes zones de disponibilité, qui se trouvent dans des sous-réseaux différents.
- L'application critique s'exécute sur tous les serveurs.
- Les utilisateurs sont connectés à une adresse IP virtuelle gérée par l'équilibreur de charge du cloud.
- SafeKit fournit un « health check » configuré dans l'équilibreur de charge. Le « health check » renvoie OK sur tous les serveurs exécutant l'application.
- En cas de défaillance ou d'arrêt d'un serveur, le « health check » ne renvoie rien à l'équilibreur de charge, qui arrête alors d'acheminer les requêtes vers ce serveur.
- SafeKit surveille l'application critique sur tous les serveurs à l'aide des checkers de SafeKit.
- SafeKit redémarre automatiquement l'application critique sur un serveur en cas de défaillance logicielle, grâce à des scripts de redémarrage.

Pour plus d'informations, consultez le cluster ferme dans Azure, le cluster ferme dans AWS ou le cluster ferme dans GCP

2. Installation

2.1 Installation de SafeKit

2.1.1 Télécharger le package

1. Se connecter à `https://support.evidian.com/safekit`
2. Aller dans `<Version 8.2>/Platforms/<Your platform>/Current versions`
3. Télécharger le package

   En Windows, deux packages sont disponibles :

   - un package Windows Installer (`safekit_windows_x86_64_8_2_x_y.msi`)
     Il dépend du runtime C VS2022 qui doit être préalablement installé
   - un bundle exécutable autonome (`safekit_windows_x86_64_8_2_x_y.exe`) qui contient l'installation SafeKit *et* le runtime C VS2022

   Choisir l'un ou l'autre package suivant que le runtime C VS2022 est installé ou non.

2.1.2 Répertoires d'installation et espace disque

SafeKit est installé dans :

`SAFE`	• sur Windows `SAFE=C:\safekit` si `%SYSTEMDRIVE%`=C: • sur Linux `SAFE=/opt/safekit`	Espace disque libre au minimum : 97MB
`SAFEVAR`	• sur Windows `SAFEVAR= C:\safekit\var` si `%SYSTEMDRIVE%`=C: • sur Linux `SAFEVAR=/var/safekit`	Espace disque libre minimum : 20MB + au moins 20MB (jusqu'à 3 GB) par module pour les dumps

2.1.3 Procédure d'installation de SafeKit

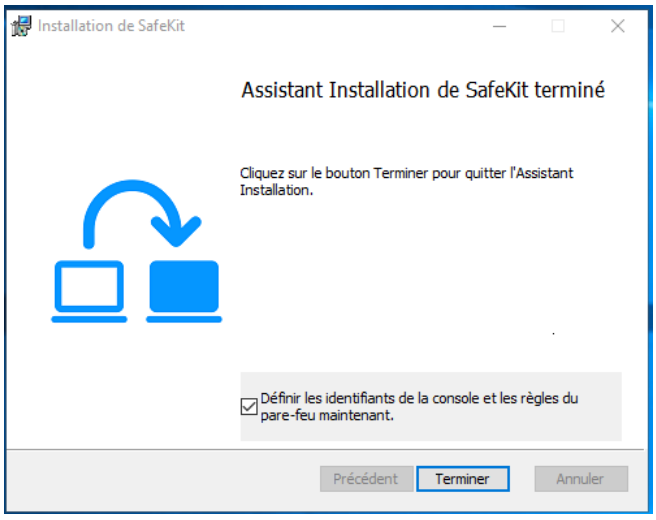
2.1.3.1 Installation sur Windows en tant qu'Administrateur

2.1.3.1.1 Installation du package SafeKit

1. Se loguer en tant qu'administrateur sur le serveur Windows
2. Localiser le fichier téléchargé safekit_windows_x86_64_8_2_x_y.msi (ou safekit_windows_x86_64_8_2_x_y.exe)
3. Installer en mode interactif en double-cliquant dessus puis dérouler l'assistant d'installation.

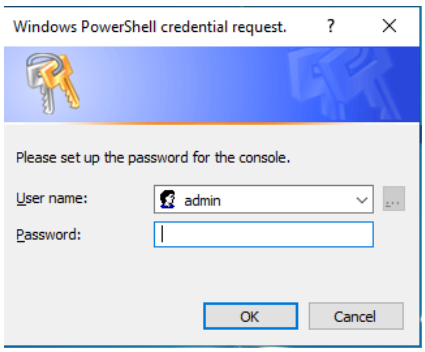
   Avant SafeKit 8.2.3, après l'installation, vous devez exécuter les scripts de configuration du pare-feu (voir la section 10.3) et d'initialisation du service web SafeKit (voir la section 11.2.1.2).

   Depuis SafeKit 8.2.3, à la fin de l'installation, il vous est demandé de cocher ou décocher « Définir les identifiants de la console et les règles du pare-feu maintenant. ».



   Si la case reste cochée, lorsque le bouton « Terminer » est cliqué :

   - Le pare-feu Microsoft est configuré pour SafeKit. Pour plus de détails ou d'autres pare-feu, voir la section 10.3.
   - Une fenêtre est ouverte pour entre le mot de passe de l'utilisateur `admin` de la console web de SafeKit.



   Cette étape est obligatoire pour initialiser la configuration par défaut du service web qui nécessite de s'authentifier. Elle est initialisée avec l'utilisateur `admin` et le

mot de passe saisi, par exemple *pwd*. Cela permet ensuite d'accéder à toutes les fonctionnalités de la console web, en se connectant avec `admin/pwd`, et d'exécuter des commandes distribuées. Pour plus de détails, voir la section 11.2.1.

Le mot de passe doit être identique sur tous les nœuds qui appartiennent au même cluster SafeKit. Sinon, la console web et les commandes distribuées échoueront avec des erreurs d'authentification.

Ou

3. Installer le .msi en mode non interactif en exécutant dans un terminal PowerShell en tant qu'administrateur:

   ```
   msiexec /qn /i safekitwindows_8_2_x_y.msi
   ```

   Une fois installé, la configuration du pare-feu et l'initialisation du service web SafeKit doivent être effectués.

2.1.3.1.2 Paramétrage du pare-feu

Cette étape est obligatoire pour permettre les communications entre les nœuds du cluster SafeKit, et avec la console web.

Aucune action supplémentaire n'est requise lorsque la configuration automatique du pare-feu a été appliquée pendant l'installation du package >= 8.2.3. Sinon, voir la section 10.3.

2.1.3.1.3 Initialisation du service web SafeKit

Cette initialisation est nécessaire pour la console web et les commandes distribuées.

Aucune action requise si l'initialisation a été faite pendant l'installation du package >= 8.2.3. Sinon, voir la section 11.2.1.2.

2.1.3.1.4 Paramétrage des antivirus

Cette étape est nécessaire uniquement en cas d'interférence de l'antivirus du serveur sur le fonctionnement de SafeKit. Voir la section 10.5 pour la liste des répertoires et processus légitimes de SafeKit qui ne doivent pas être impactés par l'antivirus.

2.1.3.2 Installation sur Linux en tant que root

2.1.3.2.1 Installation du package SafeKit

1. Se loguer en tant que root sur le serveur Linux
2. Localiser le fichier téléchargé `safekitlinux_x86_64_8_2_x_y.bin`
3. Exécuter `chmod +x safekitlinux_x86_64_8_2_x_y.bin`
4. Exécuter `./safekitlinux_x86_64_8_2_x_y.bin`

   Cela extrait le package SafeKit et le script `safekitinstall`
5. Installer en mode interactif en exécutant ./`safekitinstall`

   Pendant l'installation :

   - Répondre à "`Do you accept that SafeKit automatically configure the local firewall to open these ports (yes|no)?`"

Si vous répondez `yes`, le pare-feu Linux `firewalld` ou `iptable` est configuré pour SafeKit. Pour plus de détails ou d'autres pare-feu, voir la section 10.3.

- Répondre à "`Please enter a password or "no" if you want to set it later`".

  La saisie d'un mot de passe est obligatoire pour initialiser la configuration par défaut du service web. Celui-ci nécessite en effet de s'authentifier pour y accéder.

  Si vous répondez *`pwd`* par exemple, cette valeur est utilisée comme mot de passe pour l'utilisateur `admin`. Cela permet ensuite d'accéder à toutes les fonctionnalités de la console web, en se connectant avec `admin/`*`pwd`*, et d'exécuter des commandes distribuées. Pour plus de détails, voir la section 11.2.1.

Le mot de passe doit être identique sur tous les nœuds qui appartiennent au même cluster SafeKit. Sinon, la console web et les commandes distribuées échoueront avec des erreurs d'authentification.

Ou

5. Installer en mode non interactif en exécutant :

   `./safekitinstall -q`

   Ajouter l'option `-nofirewall` pour ne pas configurer le pare-feu.

   Ajouter l'option `-passwd` *`pwd`* pour initialiser l'authentification, requise par le service web (*`pwd`* est le mot de passe affecté à l'utilisateur `admin`).

   Le journal d'installation est `/tmp/safekitinstall_log`.

2.1.3.2.2 Paramétrage du pare-feu

Cette étape est obligatoire pour permettre les communications entre les nœuds du cluster SafeKit et avec la console web.

Aucune action supplémentaire n'est requise lorsque la configuration automatique du pare-feu a été appliquée pendant l'installation du package. Sinon, voir la section 10.3.

2.1.3.2.3 Initialisation du service web SafeKit

Cette initialisation est nécessaire pour la console web et les commandes distribuées.

Aucune action requise si l'initialisation a été faite pendant l'installation du package. Sinon, voir la section 11.2.1.2.

2.1.3.2.4 Paramétrage des antivirus

Cette étape est nécessaire uniquement en cas d'interférence de l'antivirus du serveur sur le fonctionnement de SafeKit. Voir la section 10.5 pour la liste des répertoires et processus légitimes de SafeKit qui ne doivent pas être impactés par l'antivirus.

2.1.4 Utilisation de la console web et de la ligne de commande SafeKit

Une fois installé, le cluster SafeKit doit être défini. Ensuite, les modules peuvent être installés, configurés et administrés. Toutes ces actions peuvent être effectuées avec la console ou l'interface en ligne de commande.

2.1.4.1 La console web SafeKit

1. Démarrer un navigateur web (Microsoft Edge, Firefox ou Chrome)

2. Le connecter à l'URL `http://host:9010` (où `host` est l'adresse IP ou le nom d'un nœud SafeKit)
3. Dans la section de login, s'identifier avec `admin` comme nom d'utilisateur et le mot de passe que vous avez donné pendant l'initialisation juste au-dessus (par exemple, *pwd*)
4. Une fois la console chargée, l'utilisateur `admin` a accès à la « Supervision » et à la « Configuration » dans la barre latérale de navigation, car il a le rôle `Admin` par défaut

Pour une description complète, voir la section 3.

2.1.4.2 La ligne de commande SafeKit

Elle repose sur la commande unique `safekit` située à la racine du répertoire d'installation de SafeKit. Presque toutes les commandes `safekit` peuvent être appliquées localement ou sur une liste de nœuds du cluster SafeKit. C'est ce qui est appelé commande globale ou distribuée.

Pour une description complète de la commande, voir la section 9.

Pour utiliser la commande `safekit` :

En Windows	1. Ouvrir une console PowerShell en tant qu'administrateur 2. Aller à la racine du répertoire d'installation de SafeKit `SAFE` (par défaut `SAFE=C:\safekit` si `%SYSTEMDRIVE%`=C:) `cd c:\safekit` 3. Exécuter `.\safekit.exe <arguments>` pour la commande locale 4. Exécuter `.\safekit.exe -H "<hosts>" <arguments>` pour la commande distribuée sur plusieurs nœuds
En Linux	1. Ouvrir une console Shell en tant que root 2. Aller à la racine du répertoire d'installation de SafeKit `SAFE` (par défaut `SAFE=/opt/safekit`) `cd /opt/safekit` 3. Exécuter `./safekit <arguments>` pour la commande locale 4. Exécuter `./safekit -H "<hosts>" <arguments>` pour la commande distribuée sur plusieurs nœuds

Par exemple, pour afficher la version (SafeKit, OS…):

- pour le nœud local

```
safekit level
```

- pour tous les nœuds configurés dans le cluster SafeKit

```
safekit -H "*" level
```

2.1.5 Clés de licence SafeKit

Les clés de licence sont déterminées et vérifiées en fonction du système d'exploitation (Windows ou Linux) et des noms d'hôtes des machines (pas le FQDN), comme renvoyé par la commande `hostname` dans une invite de commandes Windows ou un shell Linux. Elles sont livrées dans un fichier texte. Une fois le fichier de clé de licence installé, il n'est plus nécessaire d'être connecté à un serveur de licence.

- Si vous n'installez pas de fichier de clé de licence, le produit cessera de fonctionner tous les 3 jours. Cependant, il peut être redémarré pour 3 jours supplémentaires.
- Vous pouvez télécharger un fichier de clé d'essai d'un mois à l'adresse suivante : https://www.evidian.com/products/high-availability-software-for-application-clustering/high-availability-and-load-balancing-cluster-key/
- Lorsque la clé de licence expire ou est incorrecte (par exemple, système d'exploitation ou nom d'hôte erroné), le système entre dans le comportement de 3 jours.
- Après avoir passé une commande d'achat, vous obtenez un fichier de clé permanent (voir section 8.2). Le fichier de clé permanent peut être installé sans réinstaller ou arrêter le produit.
- Le fichier de clé peut contenir des clés pour plusieurs noms d'hôtes. SafeKit détectera la licence appropriée pour le bon système d'exploitation/nouveau d'hôte sur chaque serveur.
- Enregistrez le fichier de clé dans le fichier `SAFE/conf/license.txt` (ou tout autre fichier dans `SAFE/conf`) sur chaque serveur.
- Si les fichiers dans `SAFE/conf` contiennent plus d'un fichier de clé, la clé la plus favorable sera choisie.
- Vérifiez la conformité de la clé sur chaque serveur avec la commande `SAFE/safekit level` ou avec la console web SafeKit.

2.1.6 Caractéristiques spécifiques à chaque OS

2.1.6.1 Windows

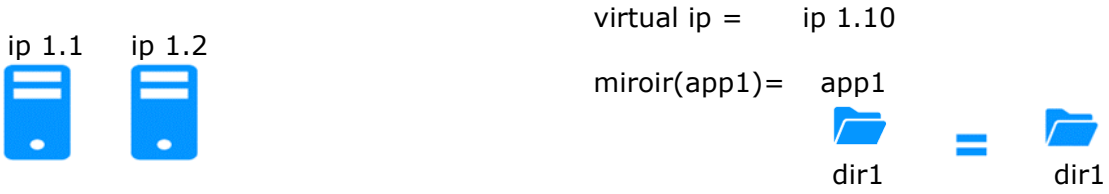
- Il faut appliquer une procédure spéciale pour arrêter proprement les modules SafeKit au shutdown d'une machine et démarrer le service `safeadmin` au boot (voir la section 10.4)
- En cas d'interfaces réseau en teaming avec du load balancing SafeKit, il est nécessaire de décocher "Vip" sur les interfaces réseau physiques de teaming et de le conserver coché seulement sur l'interface virtuelle de teaming

2.1.6.2 Linux

- En Linux, le package SafeKit dépend d'autres packages système. La plupart sont installés automatiquement, excepté ceux spécifiques à la mise en œuvre du load balancing dans une ferme et de la réplication de fichiers dans un miroir.
- Pour la liste à jour des packages nécessaires, voir le *SafeKit Release Notes*.
- L'utilisateur `safekit` et un groupe `safekit` sont créés : tous les utilisateurs appartenant au groupe `safekit` et l'utilisateur `root` peuvent exécuter des commandes SafeKit.

- Dans une ferme avec load balancing, le module kernel vip est compilé au moment de la configuration du module. Pour réussir la compilation, des packages Linux doivent être installés. Voir le *SafeKit Release Notes* pour une liste des packages à jour.
- En ferme avec load balancing sur une interface de bonding, pas d'ARP dans la configuration de bonding. Sinon l'association <adresse IP virtuelle, adresse MAC virtuelle invisible> est cassée dans les caches ARP des clients avec l'adresse MAC physique de la carte de bonding.
- En mode miroir, si utilisation de la réplication de fichiers, installer le package `nfs-util` et retirer le package `logwatch` (`rpm -e logwatch`) ; sinon le service NFS et SafeKit seront arrêtés toutes les nuits

2.2 Recommandation pour une installation d'un module miroir



2.2.1 Prérequis matériel

- au moins 2 serveurs avec le même système d'exploitation
- OS supportés : `https://support.evidian.com/supported_versions/#safekit`
- Contrôleur disque avec cache write-back recommandé pour la performance des IO

2.2.2 Prérequis réseau

- 1 adresse IP physique par serveur (ip 1.1 et ip 1.2)
- Si vous devez définir une adresse IP virtuelle (ip 1.10), les deux serveurs doivent appartenir au même réseau IP avec la configuration standard de SafeKit (LAN ou LAN étendu entre deux salles informatiques distantes). Dans le cas contraire, voir une alternative dans la section 13.5.3.

2.2.3 Prérequis application

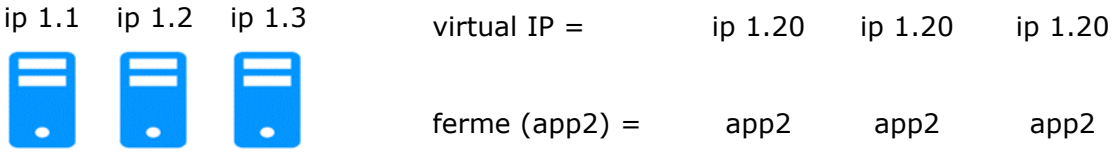
- L'application est installée et démarre sur les 2 serveurs
- L'application fournit des commandes en ligne pour démarrer et s'arrêter
- Sur Linux, commandes du style : `service "service" start|stop` ou `su -user "appli-cmd"`
- Sur Windows, commandes du style : `net start|stop "service"`
- Si nécessaire, définir une procédure de reprise suite à un crash serveur
- Retirer le démarrage automatique au boot de l'application et le remplacer par la configuration du démarrage au boot du module SafeKit

2.2.4 Prérequis réplication de fichiers

- Les répertoires de fichiers qui seront répliqués sont créés sur les 2 serveurs
- Ils se situent au même endroit sur les 2 serveurs dans l'arborescence fichier

- Il vaut mieux synchroniser les horloges des 2 serveurs pour la réplication de fichiers (protocole NTP)
- Sous Linux, aligner les valeurs des uids/gids sur les 2 serveurs pour les propriétaires des répertoires et fichiers à répliquer
- Voir aussi la section 2.1.6

2.3 Recommandation pour une installation d'un module ferme



2.3.1 Prérequis matériel

- au moins 2 serveurs avec le même OS
- OS supportés : https://support.evidian.com/supported_versions/#safekit
- Linux : outils de compilation du kernel installés pour le module kernel vip

2.3.2 Prérequis réseau

- 1 adresse IP physique par serveur (ip 1.1, ip 1.2, ip 1.3)
- Si vous devez définir une adresse IP virtuelle (ip 1.20), les serveurs doivent appartenir au même réseau IP avec la configuration standard de SafeKit (LAN ou LAN étendu entre les salles informatiques distantes). Dans le cas contraire, voir une alternative décrite dans la section 13.5.3
- voir aussi la section 2.1.6

2.3.3 Prérequis application

Les mêmes prérequis que pour un module miroir décrits en section 2.2.3.

2.4 Upgrade de SafeKit

Si vous rencontrez un problème avec SafeKit, consulter le *Software Release Bulletin* pour consulter la liste des fixs produits.

Si vous souhaitez profiter de nouvelles fonctionnalités, consulter le *SafeKit Release Notes*. Ce document vous indiquera également si vous êtes dans le cas d'un upgrade majeur (ex. 7.5 vers 8.2) qui nécessite d'effectuer une procédure différente de celle présentée ici.

La procédure d'upgrade consiste à désinstaller l'ancien package puis à réinstaller le nouveau package. Tous les nœuds du même cluster doivent être upgradé.

2.4.1 Préparer l'upgrade

1. Noter l'état "on" ou "off" des services et modules SafeKit démarrés automatiquement au boot

`safekit boot webstatus ; safekit boot status -m AM` (où *AM* est le nom du module) et en Windows : `safekit boot snmpstatus`

Le démarrage au boot du module peut être défini dans son fichier de configuration. Si c'est le cas, l'usage de la commande `safekit boot` devient inutile.

2. Pour un module miroir

   Noter le serveur qui est dans l'état `ALONE` ou `PRIM` afin de connaître le serveur avec les fichiers répliqués à jour

3. Prise de snapshots, facultative

   La désinstallation/réinstallation va réinitialiser les logs SafeKit et effacer les dumps de chaque module. Si vous souhaitez conserver ces informations, exécuter la commande `safekit snapshot -m AM /chemin/snapshot_xx.zip` pour chaque module (où *AM* est le nom du module)

2.4.2 Procédure de désinstallation

Sur Windows en tant qu'administrateur et sur Linux en tant que root :

1. Arrêter tous les modules avec la commande `safekit shutdown`

   Pour un module miroir dans l'état `PRIM-SECOND`, commencer par l'arrêt du serveur `SECOND` afin d'éviter un basculement inutile

2. Fermer tous les éditeurs, explorateur de fichiers, shells ou terminaux sous `SAFE` et `SAFEVAR`

3. Désinstaller le package SafeKit

In Windows	Désinstaller via Control Panel-Add/Remove Programs applet
In Linux	Utiliser la commande s`afekit uninstall`

4. Défaire les modifications manuelles effectuées sur le pare-feu

   Voir la section 10.3

La désinstallation de SafeKit inclut la création d'un backup des modules installés dans `SAFE/Application_Modules/backup`, puis leur déconfiguration.

2.4.3 Procédure de réinstallation et post-installation pour l'upgrade

1. Installer le nouveau package comme décrit dans la section 2.1
2. Vérifier avec la commande `safekit level` la version SafeKit installée et la validité de la licence qui n'a pas été désinstallée
3. Si vous avez un problème avec le nouveau package et l'ancienne clé, prendre une licence temporaire (voir section 2.1.5)
4. Si la console web est utilisée, vider le cache du navigateur web et forcer l'actualisation des pages HTML
5. Depuis SafeKit 8.2.1, les modules précédemment configurés sont automatiquement reconfigurés sur upgrade.

Cependant, vous pouvez avoir à reconfigurer quand même le module pour appliquer d'éventuelles évolutions de configuration apportées par la nouvelle version (consulter le *SafeKit Release Notes*). Reconfigurer le module soit avec :

- la console web en naviguant sur « Configuration/Configuration des modules/ Configurer le module/ »
- la console web en entrant directement l'URL http://host:9010/console/fr/configuration/modules/*AM*/config/
- la commande safekit config -m *AM*
- où *AM* est le nom du module

6. Reconfigurer le démarrage automatique du module au boot si nécessaire
7. Le démarrage du module au boot peut être défini dans son fichier de configuration. Si c'est le cas, passer cette étape. Si non, exécuter la commande `safekit boot -m AM on` (où *AM* est le nom du module)
8. Redémarrer les modules

Module miroir	Le module doit être démarré en primaire sur le nœud ayant les fichiers répliqués à jour (ancien `PRIM` ou `ALONE`) : • Avec la console web en naviguant sur « Supervision/ ••• du nœud/Forcer le démarrage/En primaire » • Avec la commande `safekit prim -m AM` (remplacer *AM* par le nom du module) Vérifier que l'application fonctionne correctement une fois le module dans l'état `ALONE` avant de démarrer l'autre nœud. Sur l'autre nœud (ancien `SECOND`), le module doit être démarré en secondaire : • Avec la console web en naviguant sur « Supervision/ ••• du nœud/Forcer le démarrage/En secondaire » • Avec la commande `safekit second -m AM` (remplacer *AM* par le nom du module) Une fois ce premier démarrage réalisé en sélectionnant les nœuds primaire et secondaire, les démarrages suivants peuvent être effectués avec : • Avec la console web en naviguant sur « Supervision/ ••• du nœud/ Démarrer/ » • Avec la commande `safekit start -m AM` (remplacer *AM* par le nom du module)
Module ferme	Démarrer le module soit avec : • Avec la console web en naviguant sur « Supervision/ ••• du nœud/ Démarrer/ » • Avec la commande `safekit start -m AM` (remplacer *AM* par le nom du module)

De plus, dans les cas exceptionnels où vous aviez modifié la configuration par défaut du service web SafeKit ou de la surveillance SNMP.

1. Le service web de SafeKit `safewebserver`
   - Si son démarrage automatique avait été désactivé, désactivez-le à nouveau avec la commande `safekit boot weboff`
   - Si vous aviez modifié des fichiers de configuration et que ceux-ci ont évolué dans la nouvelle version, vos modifications ont été sauvegardées dans `SAFE/web/conf` avant d'être écrasées par la nouvelle version. Le report de votre ancienne configuration dans la nouvelle version peut nécessiter quelques adaptations. Pour plus de détails sur la configuration par défaut et toutes les configurations prédéfinies, voir la section 11.

     Pour les configurations HTTPS et login/mot de passe, les certificats et les fichiers `user.conf/group.conf` générés pour la version précédente devraient être compatibles.
2. La surveillance SNMP de SafeKit
   - En Windows, si son démarrage automatique avait été activé, réactivez-le à nouveau avec la commande `safekit boot snmpon`
   - Si vous aviez modifié des fichiers de configuration et que ceux-ci ont évolué dans la nouvelle version, vos modifications ont été sauvegardées dans `SAFE/snmp/conf` avant d'être écrasées par la nouvelle version. Le report de votre ancienne configuration dans la nouvelle version peut nécessiter quelques adaptations. Pour plus de détails, voir la section 10.9.

2.5 Désinstallation complète de SafeKit

Suivre la procédure décrite ci-dessous pour désinstaller complètement SafeKit.

2.5.1 Désinstallation sur Windows

1. Se loguer en tant qu'administrateur sur le serveur Windows
2. Arrêter tous les modules à l'aide de la commande `safekit shutdown`
3. Fermer tous les éditeurs, explorateur de fichiers, ou terminaux sous `SAFE` et `SAFEVAR` (`SAFE=C:\safekit` si `%SYSTEMDRIVE%=C: ; SAFEVAR=C:\safekit\var` si `%SYSTEMDRIVE%=C:`)
4. Désinstaller le package SafeKit via Control Panel-Add/Remove Programs
5. Redémarrer le serveur
6. Détruire le répertoire `SAFE` qui correspond à l'installation précédente de SafeKit
7. Défaire les modifications effectuées pour configurer le démarrage au boot/l'arrêt au shutdown de SafeKit

   Voir la section 10.4
8. Défaire les modifications manuelles effectuées sur le pare-feu

   Voir la section 10.3

2.5.2 Désinstallation sur Linux

1. Se loguer en tant que root sur le serveur Linux

2. Arrêter tous les modules à l'aide de la commande `safekit shutdown`
3. Fermer tous les éditeurs, explorateur de fichiers, ou terminaux sous `SAFE` et `SAFEVAR` (`SAFE=/opt/safekit ; SAFEVAR=/var/safekit`)
4. Désinstaller SafeKit avec la commande `safekit uninstall -all` et répondre `yes` lorsque cela est demandé pour confirmer la destruction de tous les répertoires créés lors de la précédente installation
5. Redémarrer le serveur
6. Défaire les modifications effectuées pour paramétrer les règles de pare-feu

   Voir la section 10.3
7. Supprimer, l'utilisateur/groupe créés par l'installation précédente (par défaut `safekit/safekit`) avec les commandes :

   ```
   userdel safekit
   groupdel safekit
   ```

2.6 Documentations SafeKit

SafeKit Solution	La solution SafeKit y est entièrement détaillée.
Formation SafeKit	Reportez-vous à cette formation en ligne pour un démarrage rapide de l'utilisation de SafeKit.
SafeKit Release Notes	Il contient : • Dernières instructions d'installation • Changements majeurs • Restrictions et problèmes connus • Instructions de migration
Software Release Bulletin	Bulletin listant les packages SafeKit 8.2 avec la description des changements et des problèmes corrigés.
SafeKit Knowledge Base	Liste des problèmes et restrictions connus de SafeKit. D'autres KB sont accessibles sur le site support Evidian, mais sont accessibles uniquement aux utilisateurs enregistrés. Pour plus de détails sur le site support, voir la section 8.
Guide de l'utilisateur SafeKit	Il s'agit de ce guide. Veillez à consulter le guide correspondant à votre numéro de version SafeKit. Celui-ci est installé avec le package SafeKit et accessible via la console web sous ⋮/? Guide de l'utilisateur. Le lien ci-contre renvoie à la dernière version de ce guide.

3. La console web de SafeKit

La console web et l'API SafeKit ont évolué en SafeKit 8 par rapport aux versions précédentes. Par conséquent, la console livrée avec SafeKit 8 n'est capable d'administrer que des serveurs avec SafeKit 8 ; de plus, ceux-ci ne peuvent être administrés avec une ancienne console.

3.1 Démarrer la console web

La console web de SafeKit offre la possibilité d'administrer un cluster SafeKit. Un cluster SafeKit est un groupe de serveurs sur lesquels Safekit est installé et fonctionnel. Tous les serveurs appartenant à un cluster donné partagent la même configuration de cluster (liste des serveurs et réseaux utilisés) et communiquent entre eux afin d'avoir une vue globale des configurations des modules installés. Un même serveur ne peut appartenir à plusieurs clusters.

3.1.1 Lancer un navigateur web

- Le navigateur web peut être lancé sur n'importe quelle station de travail ou serveur ayant un accès réseau au(x) serveur(s) SafeKit et ayant l'autorisation d'accès
- Les réseaux, pare-feu et proxy doivent être configurés de manière à permettre l'accès à tous les serveurs SafeKit
- Le navigateur doit autoriser l'exécution de Javascript
- La console web a été validée avec les navigateurs Microsoft Edge, Firefox et Chrome
- Pour éviter les pop-ups de sécurité avec Microsoft Edge, il faut ajouter les adresses des serveurs SafeKit dans la zone des sites de confiance ou la zone d'Intranet local du navigateur
- Les messages dans la console sont affichés en anglais, français en fonction de la langue sélectionnée depuis la console
- Après upgrade de SafeKit, il est nécessaire de vider le cache du navigateur de façon à recharger la nouvelle console web. Pour cela, vous pouvez utiliser un raccourci clavier :

  1. Ouvrir le navigateur sur n'importe quelle page web, et presser en même temps les touches `Ctrl`, `Shift` et `Suppr`
  2. Cela ouvre une fenêtre de dialogue : cocher tous les items puis cliquer le bouton Nettoyer maintenant ou Supprimer.
  3. Fermer le navigateur, arrêter tous les processus du navigateur qui continueraient à tourner en tâche de fond et le relancer pour la charger la console web

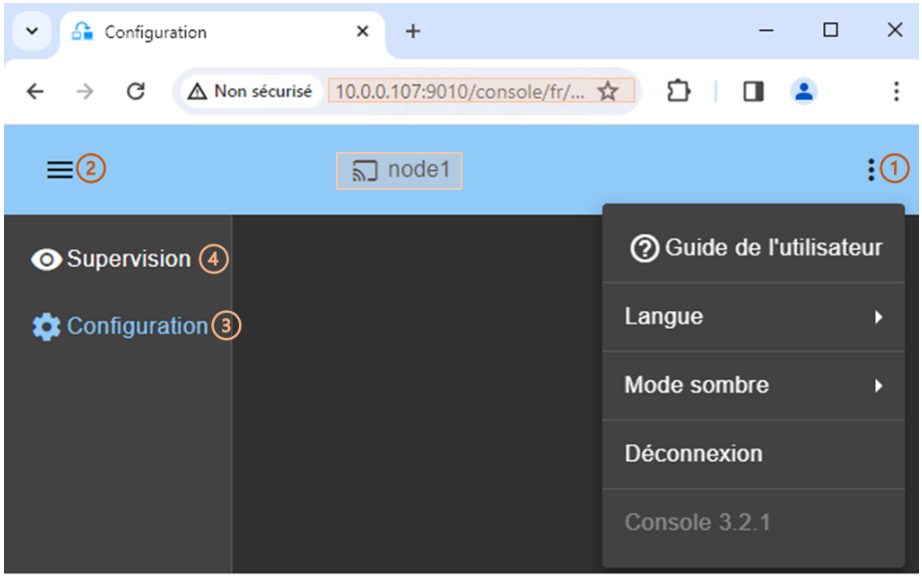
3.1.2 Connecter la console à un serveur SafeKit

Par défaut, l'accès à la console web nécessite que l'utilisateur s'authentifie avec un nom et mot de passe. A l'installation de SafeKit, vous avez dû l'initialiser avec l'utilisateur `admin` et lui affecter un mot de passe. Ce nom `admin`, et ce mot de passe sont suffisants pour accéder à toutes les fonctionnalités de la console. Pour plus de détails sur cette configuration, voir la section 11.2.1.

1. Lancer un navigateur web (Microsoft Edge, Firefox, or Chrome)
2. Le connecter à l'URL `http://host:9010` (`host` est le nom ou l'adresse IP d'un des serveur SafeKit). Si HTTPS est configuré, il y a une redirection automatique sur `https://host:9453`.
3. Le serveur SafeKit sur laquelle la console est connectée (`host` dans l'URL) est nommé nœud de connexion. Ce nœud agit en tant que proxy pour communiquer au compte de la console avec tous les autres serveurs SafeKit.

Vous pouvez vous connecter à n'importe quel nœud du cluster puisque la console offre une vue et des actions globales. En cas d'erreur de connexion avec un nœud, connectez-vous à un autre nœud.

4. Dans la page de login, s'identifier avec `admin` comme nom d'utilisateur et le mot de passe que vous avez donné pendant l'initialisation (par exemple, *pwd*).
5. La console web de SafeKit est chargée



- Quand la console est connectée à un serveur SafeKit sur lequel le cluster est configuré, le nom du nœud correspond au serveur (tel que défini dans la configuration du cluster) est affiché dans l'entête. Il s'agit **du nœud de connexion** (`node1` dans l'exemple).

  Si le cluster n'est pas encore configuré, aucun nom n'est affiché.

- (1) Cliquer sur ⋮ pour ouvrir le menu afin d'accéder au Guide de l'utilisateur SafeKit, sélectionner la langue, activer/désactiver le mode sombre et se déconnecter.
- (2) Cliquer sur ≡ pour réduire ou développer la barre latérale de navigation.

- (3) Cliquer sur « Configuration » pour configurer le cluster et les modules. La configuration n'est autorisée qu'aux utilisateurs ayant le rôle Admin. Par défaut, l'utilisateur `admin` a le rôle Admin.
- (4) Cliquer sur « Supervision » pour superviser et contrôler les modules configurés. La supervision est autorisée aux utilisateurs qui ont les rôles Admin, Control et Monitor. Avec le rôle Monitor, les actions sur les modules (démarrage, arrêt...) sont interdites

La console web offre des aides contextuelles en cliquant sur l'icône .

3.2 Configurer un cluster SafeKit

Le cluster SafeKit doit être défini avant d'installer, de configurer ou de démarrer un module SafeKit.

Le cluster est défini par un ensemble de réseaux et les adresses, sur ces réseaux, d'un groupe de serveurs SafeKit, appelés nœuds. Ces nœuds mettent en œuvre un ou plusieurs modules. Un serveur n'est pas obligatoirement connecté à tous les réseaux du cluster, mais tous les serveurs sont connectés à au moins un.

La configuration du cluster est sauvegardée du côté des serveurs dans le fichier `cluster.xml` (voir section 12). Pour que le fonctionnement soit correct, il est impératif que la configuration du cluster soit identique sur tous les nœuds.

Il est préférable de définir complètement tous les nœuds du cluster avant de configurer les modules. En effet, la modification de la configuration du cluster peut impacter la configuration et l'exécution des modules déjà installés.

La page d'accueil de la configuration du cluster est accessible :

- Directement via l'URL http://host:9010/console/fr/configuration/cluster

Ou

- En naviguant dans la console via « Configuration/Configuration du cluster »

Si le cluster n'est pas encore configuré, l'assistant de configuration du cluster s'ouvre automatiquement au chargement de la console web.

3.2.1 L'assistant de configuration du cluster

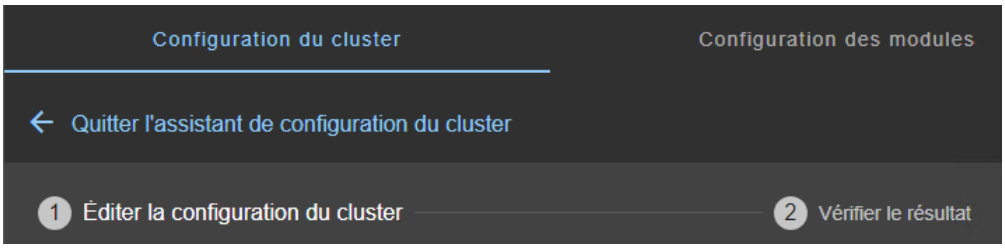
Ouvrir l'assistant de configuration :

- Directement via l'URL http://host:9010/console/fr/configuration/cluster/config

Ou

- En naviguant dans la console via « Configuration/Configuration du cluster/ Configurer le cluster »

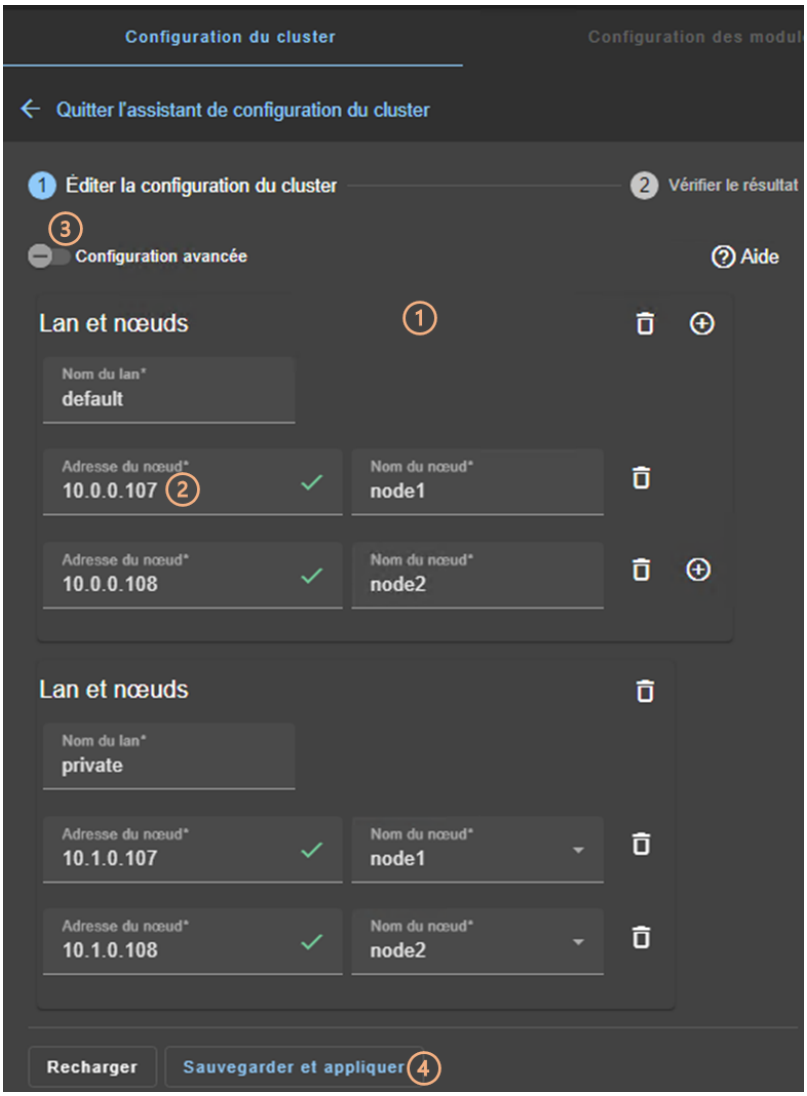
L'assistant de configuration du cluster est un formulaire à étapes :



1. « Éditer la configuration du cluster » décrit en section 3.2.1.1
2. « Vérifier le résultat » décrit en section 3.2.1.2
3. ← pour « Quitter l'assistant de configuration du cluster »

3.2.1.1 Éditer la configuration du cluster

L'assistant de configuration du cluster est un formulaire guidé étape par étape.



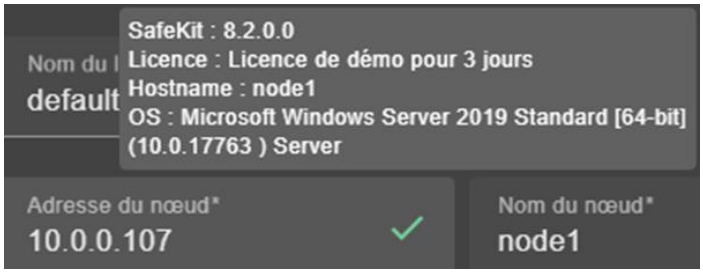
- (1) Remplir le formulaire pour affecter d'abord le nom convivial pour le réseau. Ce nom est utilisé plus tard pour configurer les réseaux de surveillance utilisés par un module.

Cliquer sur ⊕ pour ajouter un nouveau nœud /lan ou sur 🗑 pour supprimer un nœud/lan du cluster.

Quand un nœud/lan est supprimé du cluster, tous les modules l'utilisant dans sa configuration peuvent devenir inutilisables.

- (2) Saisir l'adresse IP du nœud, puis appuyer sur la touche tabulation pour vérifier la disponibilité du serveur et l'insertion automatique de son nom.

  L'icône à côté de l'adresse reflète l'accessibilité du nœud.



✓ signifie que le server SafeKit est disponible. L'infobulle donne des informations sur le serveur.

✗ signifie que le serveur n'a pas répondu dans le délai imparti. Résoudre le problème, afin de pouvoir administrer ce nœud. Cela peut être dû à une mauvaise adresse, une défaillance du réseau ou du serveur, une mauvaise configuration du navigateur web ou du pare-feu, l'arrêt du service web SafeKit sur le nœud. Pour investiguer le problème, voir la section 7.1.

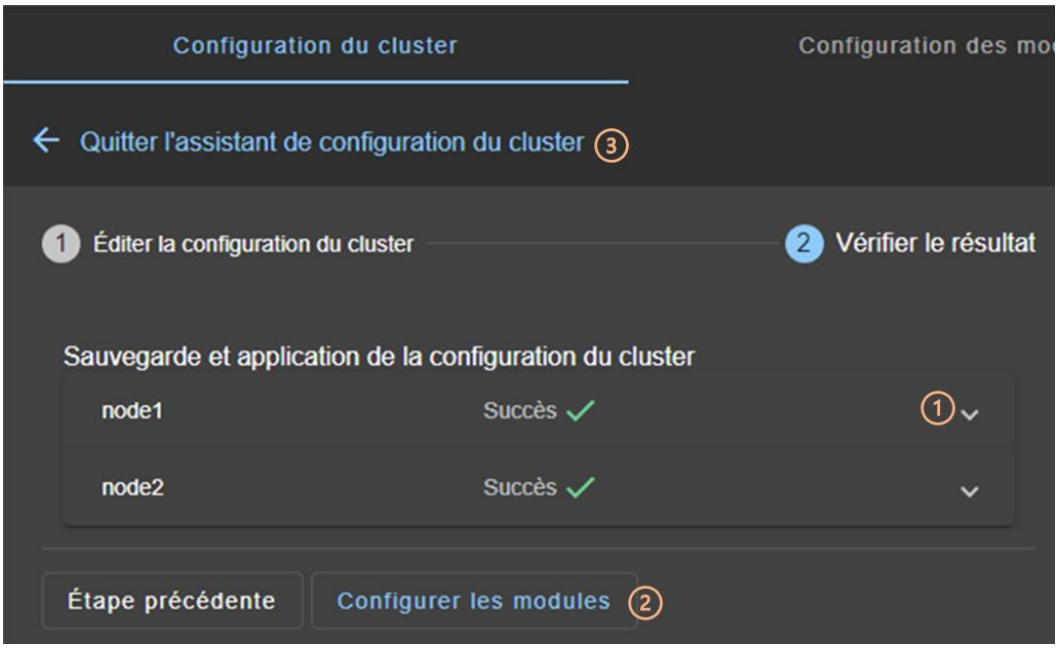
- Modifier le nom du nœud si besoin. Ce nom est celui qui sera utilisé par le service d'administration de SafeKit pour identifier de manière unique chaque nœud du cluster. C'est également le nom affiché dans la console web.
- (3) Si besoin, cliquer sur « Configuration avancée » pour basculer sur l'édition du cluster au format XML.

  Cliquer sur ⓘ pour ouvrir le Guide de l'utilisateur SafeKit sur la description de la configuration dans le fichier `cluster.xml`.
- Cliquer sur « Recharger » pour abandonner vos modifications en cours et recharger la configuration d'origine.
- (4) Une fois l'édition terminée, cliquez sur « Sauvegarder et appliquer » pour enregistrer et appliquer la configuration à tous les nœuds du cluster.

Si besoin, vous pouvez réappliquer la configuration du cluster sur tous les nœuds sans la modifier.

Pour des exemples de configuration du cluster avec deux réseaux voir la section 15.1.1; avec trois nœuds voir la section 15.2.1.

3.2.1.2 Vérifier le résultat



- (1) Lire le résultat de la configuration sur chaque nœud :
  - « Succès » ✓ signifie que la configuration a réussi.
  - « Échec » ✗, signifie que la configuration a échoué. Cliquer sur ⌄ pour lire la sortie des commandes exécutées sur le nœud et rechercher l'erreur. Vous pouvez avoir à modifier les paramètres saisis ou à vous connecter au serveur afin de corriger le problème. Une fois l'erreur corrigée, Sauvegarder et appliquer à nouveau.
- (2) Cliquer sur « Configurer les modules » pour quitter l'assistant de configuration du cluster et naviguer vers la configuration des modules.

Ou

- (3) Cliquer sur ← pour « Quitter l'assistant de configuration du cluster » et aller sur la page d'accueil de configuration du cluster.

3.2.2 Page d'accueil de la configuration du cluster

Lorsque le cluster est configuré, la page d'accueil de la configuration du cluster est accessible.

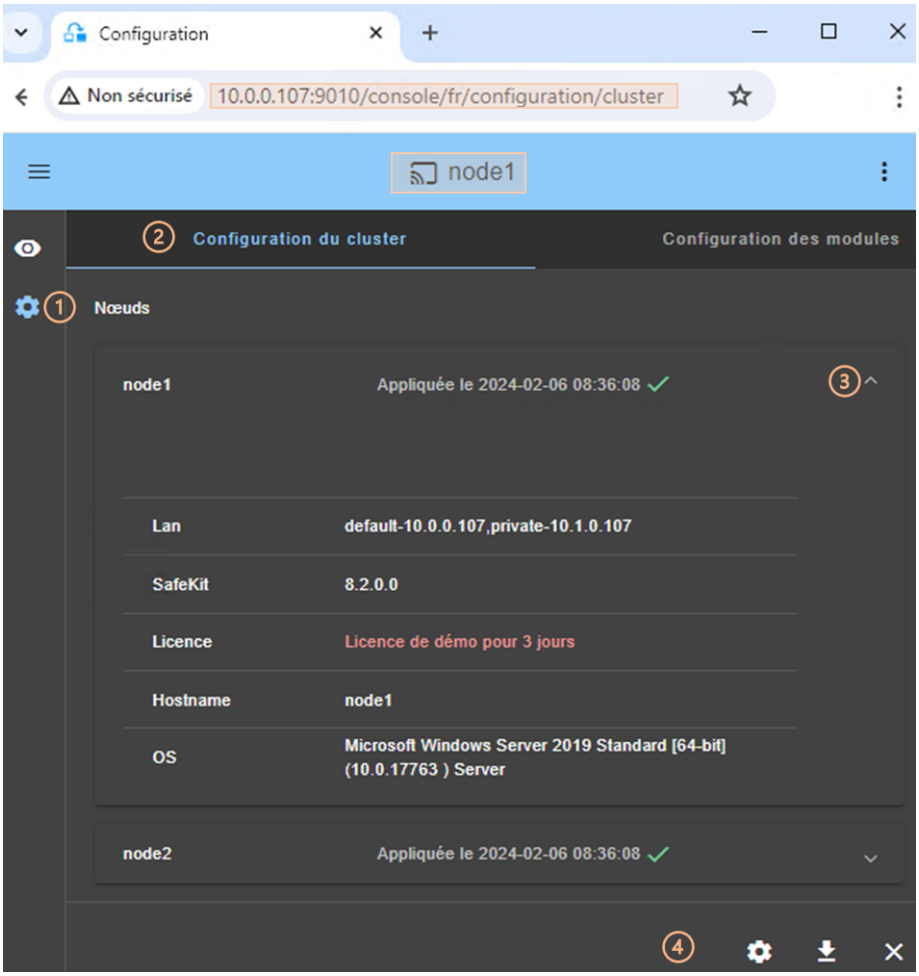
L'ouvrir :

- Directement avec http://host:9010/console/fr/configuration/cluster

Ou

- En naviguant dans la console sur ⚙ « Configuration/Configuration du cluster »

Dans cet exemple, la console est chargée depuis `10.0.0.107` qui correspond au nœud `node1` dans le cluster existant. Il s'agit du nœud de connexion.



- (1) Cliquer sur ⚙ « Configuration » dans la barre de navigation latérale
- (2) Cliquer sur l'onglet « Configuration du cluster »

  Les nœuds configurés dans le cluster sont listés avec leur date de configuration
- (3) Cliquer sur ⌄ pour afficher des détails sur le nœud.

  Il s'agit du nom des lans et adresses définies dans la configuration du cluster, version et licence SafeKit, hostname et OS.
- (4) Cliquer sur l'un des boutons :
  - ⚙ pour modifier la configuration du cluster ou réappliquer la configuration courante. Cela ouvre l'assistant de configuration du cluster et charge la configuration courante depuis le nœud de connexion.
  - 🡻 pour télécharger la configuration du cluster au format XML depuis le nœud de connexion.
  - ✕ pour déconfigurer le cluster sur un ou plusieurs nœuds.

3.3 Configurer un module

Une fois le cluster configuré, il est possible de configurer un nouveau module sur le cluster. La page d'accueil de la configuration des modules est accessible :

- Directement via l'URL http://host:9010/console/fr/configuration/modules

Ou

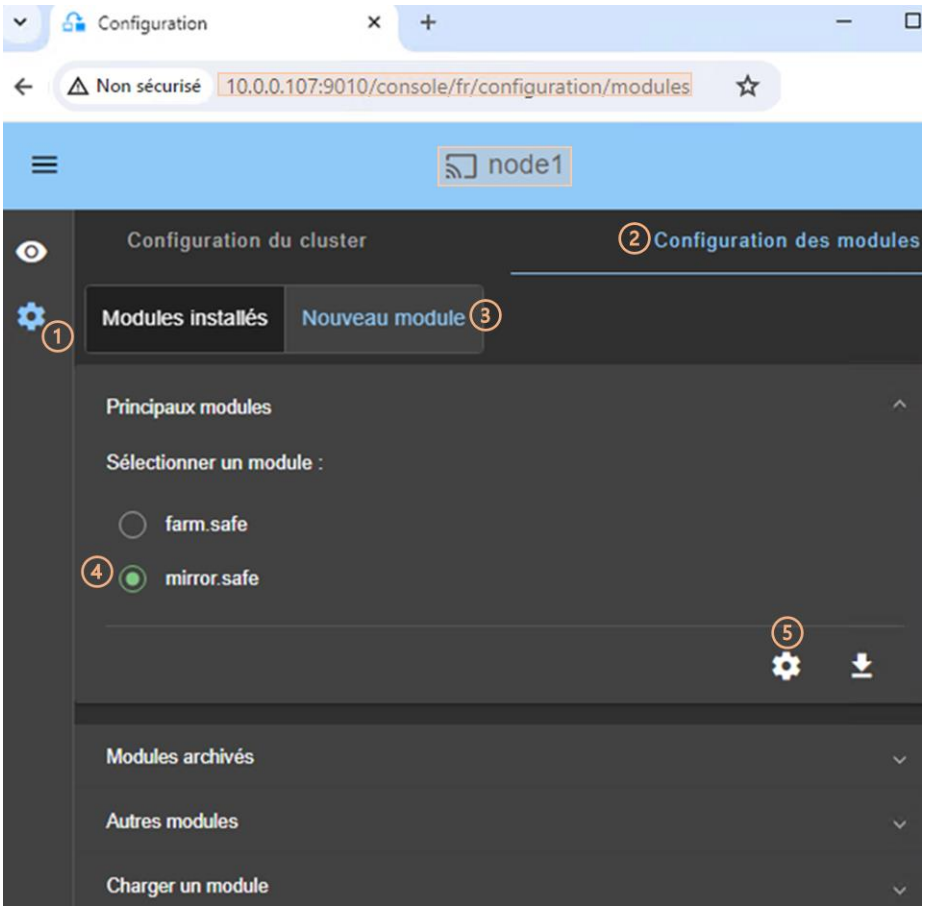
- En naviguant dans la console via ⚙ « Configuration/Configuration des modules »

S'il n'y a aucun module configuré, la console présente automatiquement la page pour configurer un « Nouveau Module ».

Pour des exemples de configuration de modules voir la section 15.

3.3.1 Sélectionner le nouveau module à configurer

Dans cet exemple, la console est chargée depuis `10.0.0.107` qui correspond au nœud `node1` dans le cluster existant. Il s'agit du nœud de connexion.



- Cliquer sur ⚙ « Configuration » dans la barre de navigation latérale
- (2) Cliquer sur l'onglet « Configuration des modules »
- (3) Cliquer sur « Nouveau module »
- La page propose de sélectionner un nouveau module parmi plusieurs propositions visibles en cliquant sur ⌄ :

- Les « Principaux modules », notamment les modules génériques `mirror.safe` (voir la section 15.1.2) et `farm.safe` (voir la section 15.2.2) à utiliser pour l'intégration d'une nouvelle application dans une architecture miroir ou ferme.

  Sont présentés les modules stockés sur le nœud de connexion, `node1`, sous `SAFE/Application_Modules/generic`, `SAFE/Application_Modules/demo` et `SAFE/Application_Modules/published`.

- Les « Modules archivés » sur le nœud de connexion qui sont sauvegardés lorsqu'un module est désinstallé sur ce nœud.

  Ils sont récupérés depuis `node1` sous `SAFE/Application_Modules/backup`.

- D'« Autres modules » qui sont des exemples d'utilisation de fonctionnalités de SafeKit dans des modules fournis en vue de tests uniquement. Voir la section 15 pour la description de certains d'entre eux.

  Ils sont récupérés depuis `node1` sous `SAFE/Application_Modules/other`.

- Un module stocké localement accessible depuis « Charger un module ».

  Cette fonctionnalité peut être utilisée pour configurer un module, pour une application donnée (par exemple, Microsoft SQL Server, PostgreSQL…), téléchargé depuis l'un des guides d'installation rapide de SafeKit.

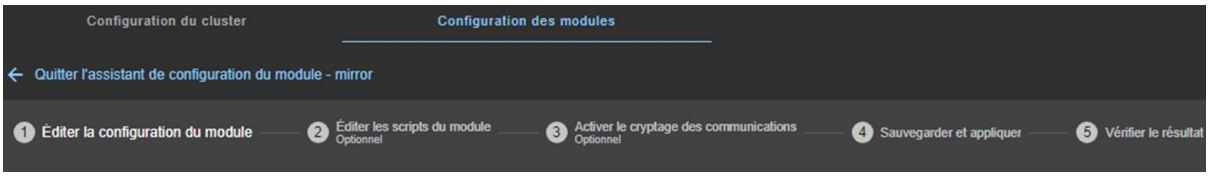
- (4) Sélectionner un module à configurer parmi les propositions listées ci-dessus. Dans l'exemple, `mirror.safe`.
- (5) Cliquer sur le bouton ⚙ « Configurer le nouveau module ».
- Un dialogue s'ouvre pour saisir le nom du nouveau module

- (6) Entrer le nom du nouveau module
- (7) Cliquer sur « Confirmer »

L'assistant de configuration du module est ouvert. Celui-ci est décrit ci-dessous.

3.3.2 L'assistant de configuration du module

L'assistant de configuration du module est un formulaire à étapes :



1. « Éditer la configuration du module » décrit en section 3.3.2.1
2. « Éditer les scripts du module (Optionnel) » décrit en section 3.3.2.2
3. « Activer le cryptage des communications (Optionnel) » décrit en section 3.3.2.3

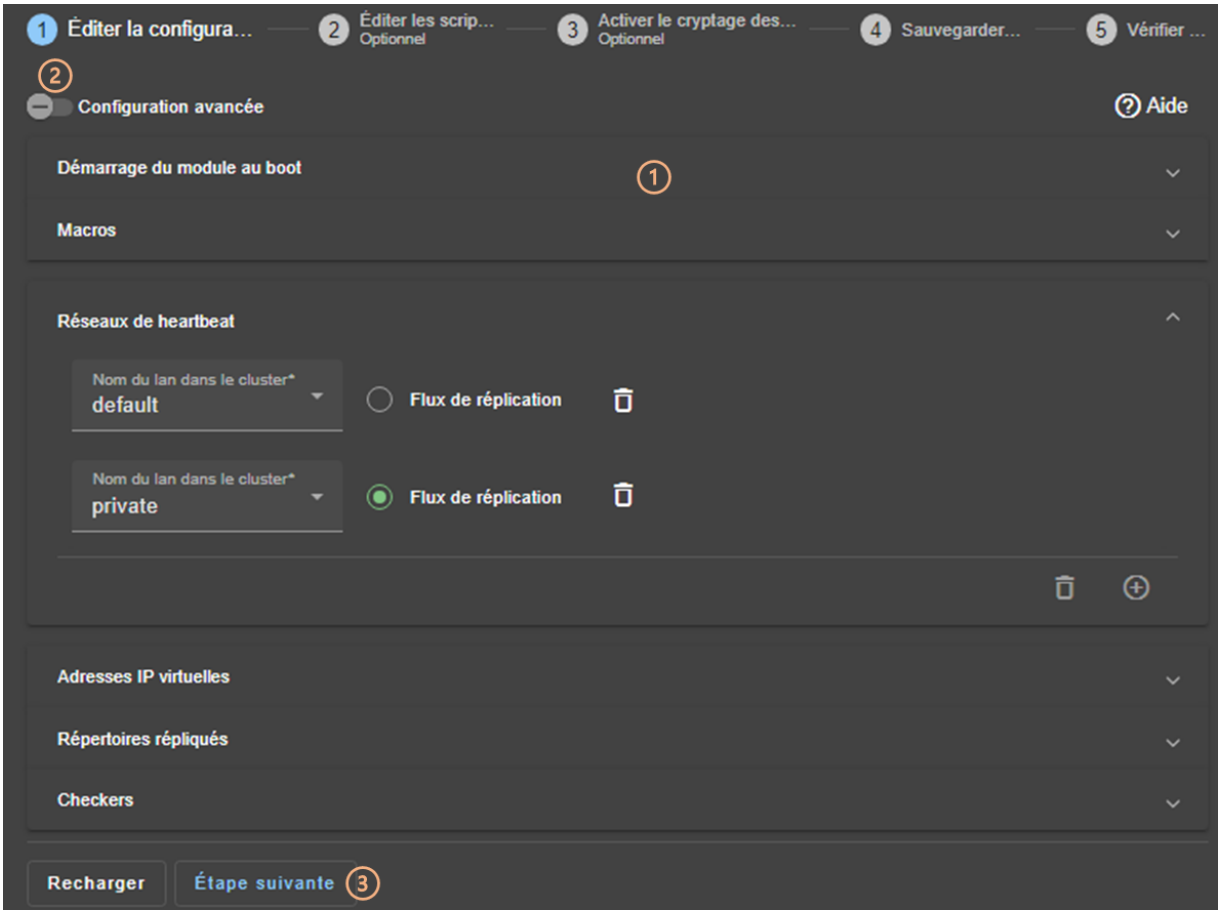
4. « Sauvegarder et appliquer » décrit en section 3.3.2.4
5. « Vérifier le résultat » décrit en section 3.3.2.5
6. ← pour « Quitter l'assistant de configuration du module »

Notez que la reconfiguration d'un module ne peut être appliquée qu'aux nœuds sur lesquels le module en question n'est pas démarré. Il convient donc d'arrêter le module avant de lancer l'assistant de configuration.

Si besoin, vous pouvez réappliquer la configuration du module sur tous les nœuds sans la modifier.

3.3.2.1 Éditer la configuration du module

Ci-dessous l'exemple de l'édition de la configuration du module miroir `mirror.safe`.



- (1) Remplir le formulaire pour affecter les valeurs aux différents composants, en ajouter ou en supprimer. Cliquer sur ⌄ pour ouvrir le panneau détaillé pour chaque composant.

  Ce formulaire permet de saisir uniquement les principaux paramètres de configuration du module.

Le nom des « Réseaux de heartbeat » proposés sont les noms des lans saisis lors de la configuration du cluster.

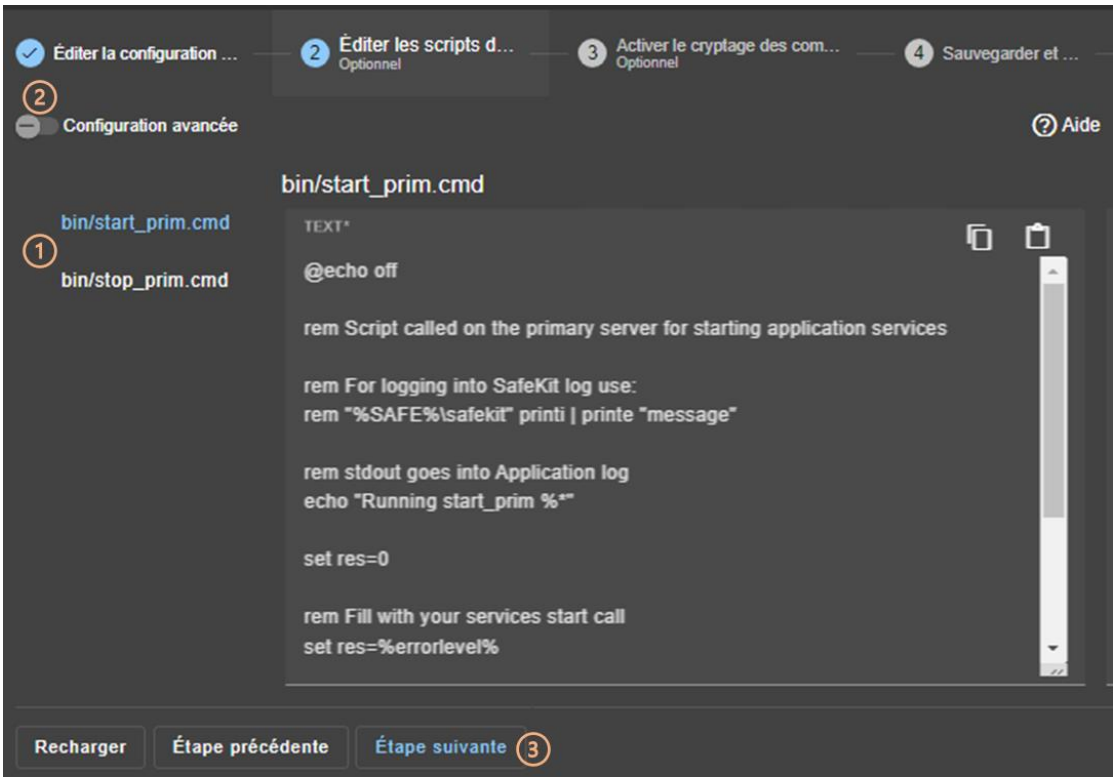
- (2) Pour une configuration avancée du module, exhaustive par rapport au formulaire, cliquer sur « Configuration avancée ». Cela bascule sur l'édition du fichier de configuration du module au format XML, `userconfig.xml`.

  Cliquer sur ⓘ pour ouvrir le Guide de l'utilisateur SafeKit sur la description de la configuration des différents composants dans le fichier `userconfig.xml`.
- Si besoin, cliquer sur « Recharger » pour abandonner vos modifications et recharger la configuration complète d'origine (y compris les scripts si ceux-ci avaient été modifiés dans l'étape suivante).
- (3) Une fois l'édition de la configuration du module achevée, cliquer sur « Étape suivante ».

Pour des exemples de configuration d'un module miroir voir la section 15.1.2 ; d'un module ferme voir la section 15.2.2.

3.3.2.2 Éditer les scripts du module

Ci-dessous l'exemple de l'édition des scripts du module miroir `mirror.safe`.



- (1) Cliquer sur « start_prim » ou « stop_prim » pour l'éditer et y insérer le démarrage/arrêt de votre application.

  Cliquer sur pour copier le contenu du script et l'éditer avec votre éditeur syntaxique favori. Une fois fait, copier le contenu mis à jour dans le champ d'input avec .
- (2) Si besoin, cliquer sur « Configuration Avancée » pour lister les autres scripts du module et les éditer (`prestart`, `poststop`, scripts pour les checkers…).

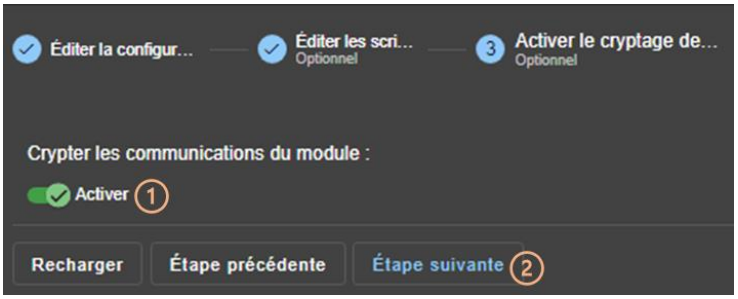
  Cliquer sur ⓘ pour ouvrir le Guide de l'utilisateur SafeKit sur la description des scripts du module.

- Si besoin, cliquer sur « Recharger » pour abandonner vos modifications et recharger la configuration complète d'origine (y compris la configuration du module si celle-ci avait été modifiée dans l'étape précédente).
- (3) Une fois l'édition des scripts du module achevée, cliquer sur « Étape suivante ».

Pour des exemples de scripts d'un module miroir voir la section 15.1.3 ; d'un module ferme voir la section 15.2.3.

3.3.2.3 Activer le cryptage des communications

Le cryptage des communications internes du module entre les nœuds du cluster, est activé par défaut. Pour plus de détails, voir la section 10.6.



- (1) Cliquer Activer pour activer ou désactiver le cryptage des communications du module.

Lorsque la clé de cryptage du module n'est pas identique sur tous les nœuds, la communication interne est impossible. Il faut réappliquer la configuration sur tous les nœuds pour propager la même clé.

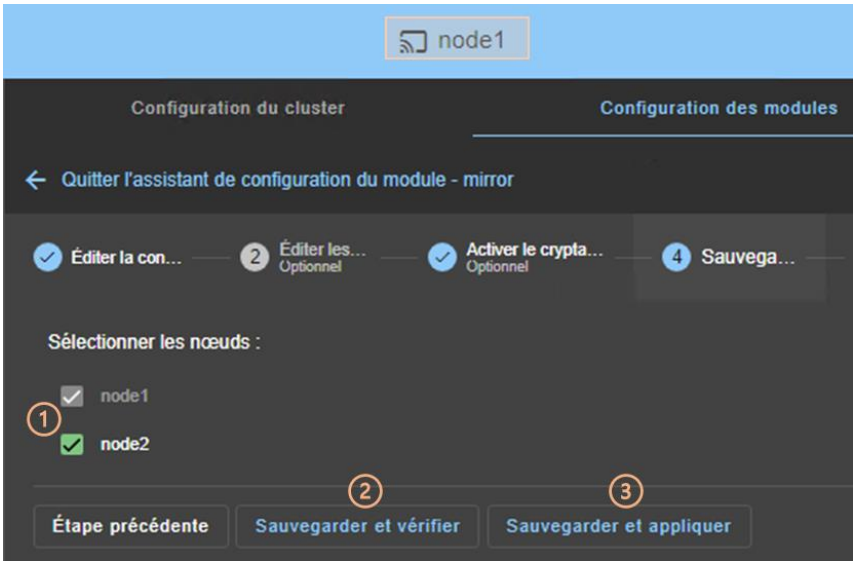
Pour générer de nouvelles clés de cryptage, il faut :

1. Désactiver le cryptage, puis « Sauvegarder et appliquer » la configuration sur tous les nœuds
2. Activer le cryptage, puis « Sauvegarder et appliquer » la configuration sur tous les nœuds

- Si besoin, cliquer sur « Recharger » pour abandonner vos modifications et recharger la configuration complète d'origine (y compris la configuration du module et les scripts si ceux-ci avaient été modifiées dans les étapes précédentes).
- (2) Une fois cette étape achevée, cliquer sur « Étape suivante ».

3.3.2.4 Sauvegarder et appliquer

Étape de sélection des nœuds concernés par la configuration.

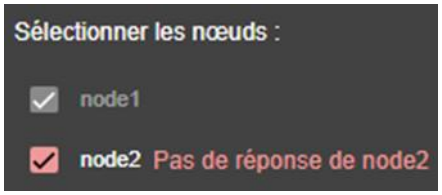


- (1) Cochez/décochez pour sélectionner/désélectionner les nœuds. Veuillez noter que le nœud de connexion (`node1` dans l'exemple) est obligatoire.

  Il y a 2 cas où « Sauvegarder et appliquer » est désactivé :

  Le module sur le nœud sélectionné est démarré et dans un état différent de ✕`STOP (NotReady)`.

  Le nœud sélectionné n'a pas répondu dans le délai imparti. Cela peut être dû à une mauvaise adresse, une défaillance du réseau ou du serveur, une mauvaise configuration du navigateur web ou du pare-feu, l'arrêt du service web SafeKit sur le nœud. Pour investiguer le problème, voir la section 7.1.

  Dans les deux cas, décochez le nœud ou cliquer sur « Sauvegarder et vérifier » pour l'appliquer plus tard, après avoir arrêté le module ou résolu le problème de communication.

- (2) Cliquer sur « Sauvegarder et vérifier » pour sauvegarder la configuration modifiée sur le nœud de connexion et vérifier sa cohérence. Il passe ensuite à l'étape suivante pour afficher le résultat de cette opération.

  Une fois cette opération terminée, toutes les modifications sont enregistrées sur le nœud de connexion. L'assistant de configuration peut être quitté et relancé ultérieurement pour appliquer la configuration sauvegardée. Tant que la configuration sauvegardée n'est pas appliquée, la dernière configuration appliquée reste active.

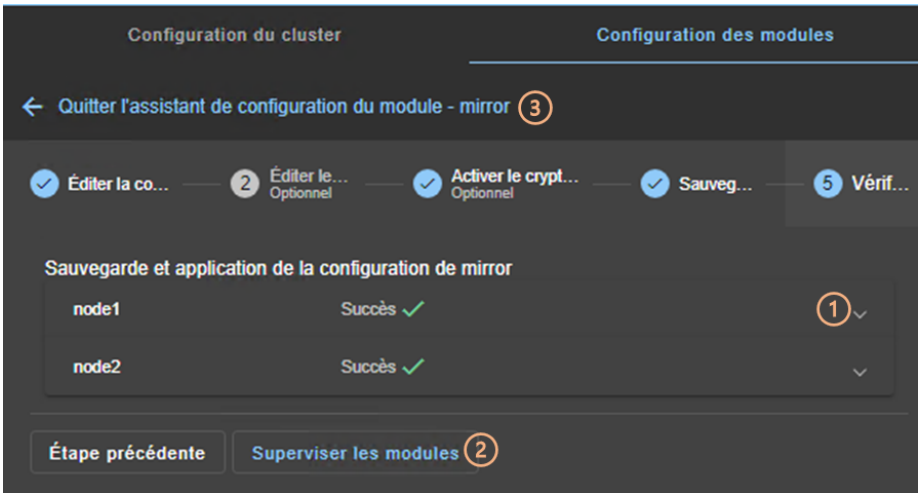
- (3) Cliquer sur « Sauvegarder et appliquer » pour sauvegarder et appliquer la configuration modifiée aux nœuds sélectionnés. Il passe ensuite à l'étape suivante pour afficher le résultat de cette opération.

Si l'opération est réussie, la configuration appliquée devient la configuration active du module.

La configuration du module est sauvegardée du côté serveur sous `SAFE/modules/AM` (où *AM* est le nom du module). Lors de la reconfiguration du module, ce répertoire est détruit et écrasé à partir des modifications faites dans la console. Du côté serveur, fermer tous les éditeurs, explorateur de fichiers, shells ou cmd sous `SAFE/modules/AM` (au risque sinon que la configuration se passe mal).

3.3.2.5 Vérifier le résultat

L'exemple ci-dessous montre le résultat de l'opération « Sauvegarder et appliquer ». La présentation pour « Sauvegarder et vérifier » est similaire.



- (1) Lire le résultat de l'opération sur chaque nœud :
  - « Succès » ✓ signifie que l'opération a réussi
  - « Échec » ✗, signifie que l'opération a échoué.

    Cliquer sur ⌄ pour lire la sortie des commandes exécutées sur le nœud et rechercher l'erreur. Vous pouvez avoir à modifier les paramètres saisis ou à vous connecter au serveur afin de corriger le problème. Une fois l'erreur corrigée, répéter l'opération depuis l'étape précédente.
- (2) Ou cliquer sur « Superviser les modules » pour quitter l'assistant de configuration du module et naviguer vers la supervision des modules.

Ou

- (3) Cliquer sur ← pour « Quitter l'assistant de configuration du module » et aller sur la page d'accueil de configuration des modules.

3.3.3 Page d'accueil de la configuration des modules

Lorsqu'un premier module est configuré, la page d'accueil de la configuration des modules est accessible. Elle permet de visualiser les modules installés sur le cluster et d'accéder à la configuration d'un nouveau module.

L'ouvrir :

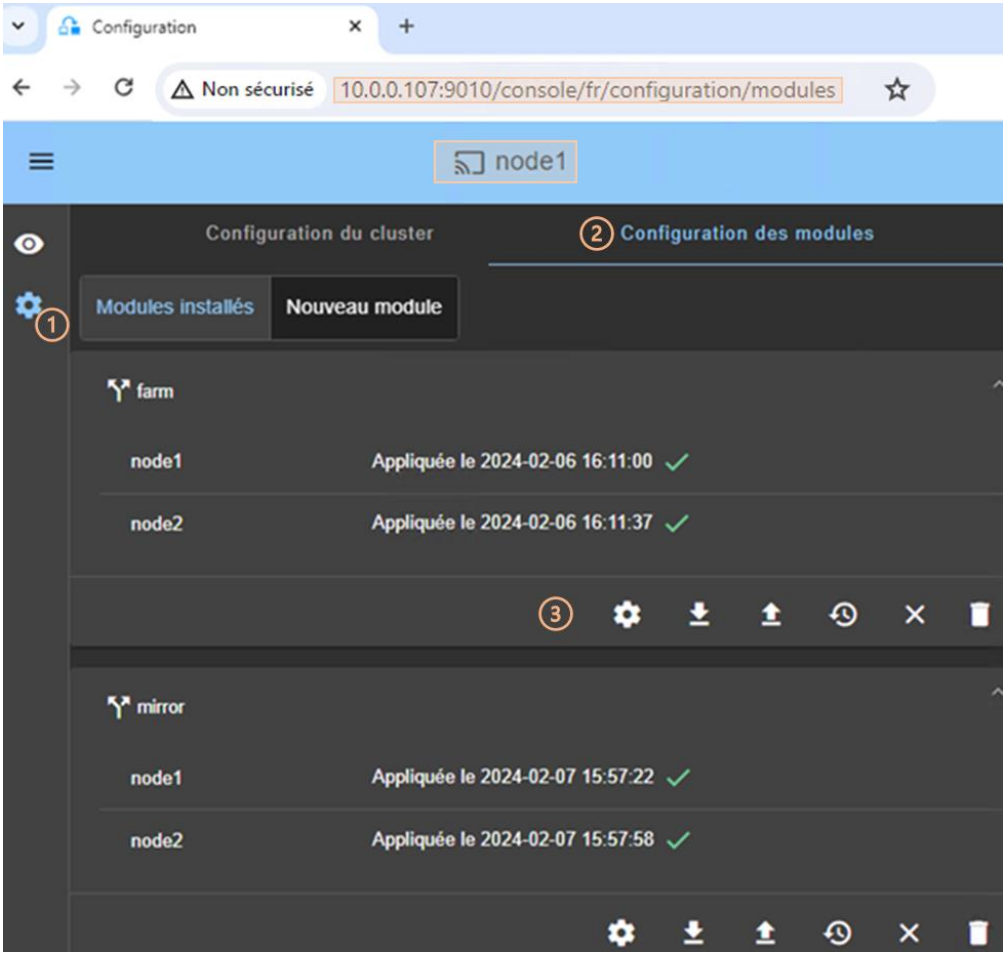
- Directement avec http://host:9010/console/fr/configuration/modules

Ou

- En naviguant dans la console sur ⚙« Configuration/Configuration des modules »

Avant chaque reconfiguration, déconfiguration et désinstallation, sur chaque nœud, fermer tous les éditeurs, explorateur de fichiers, shells ou cmd sous `SAFE/modules/AM` (au risque sinon que l'opération échoue).

Dans l'exemple suivant, la console est chargée depuis `10.0.0.107` qui correspond au nœud `node1` dans le cluster existant. Il s'agit du nœud de connexion.



- (1) Cliquer sur ⚙« Configuration » dans la barre de navigation latérale.
- (2) Cliquer sur l'onglet « Configuration des modules ».
- Les modules installés dans le cluster sont listés avec la date d'application de la configuration et éventuellement la date de sauvegarde d'une configuration qui n'a pas été encore appliquée.
- (3) Cliquer sur l'un des boutons associés au module :
  - ⚙ pour modifier sa configuration ou réappliquer sa configuration courante. Cela ouvre l'assistant de configuration du module et charge sa configuration courante depuis le nœud de connexion.

- o pour télécharger le `.safe`, composé de tous les fichiers du module (`userconfig.xml`, scripts) depuis le nœud de connexion.
- o pour reconfigurer le module depuis le contenu d'un `.safe` stocké localement.
- o pour restaurer une ancienne configuration du module.

  SafeKit conserve une copie des trois dernières configurations réussies (stockées sous `SAFE/modules/lastconfig` du côté serveur). L'ensemble des fichiers de configuration du module sont empaquetés dans un fichier `.safe`, dont le nom est du type *AM*`_<date>_<heure>` (où *AM* est le nom du module).
- o ✕ pour déconfigurer le module sur un ou plusieurs nœuds, sans le désinstaller. Ses fichiers de configuration sont conservés pour être éventuellement réappliquer plus tard.
- o pour désinstaller complètement le module sur un ou plusieurs nœuds.

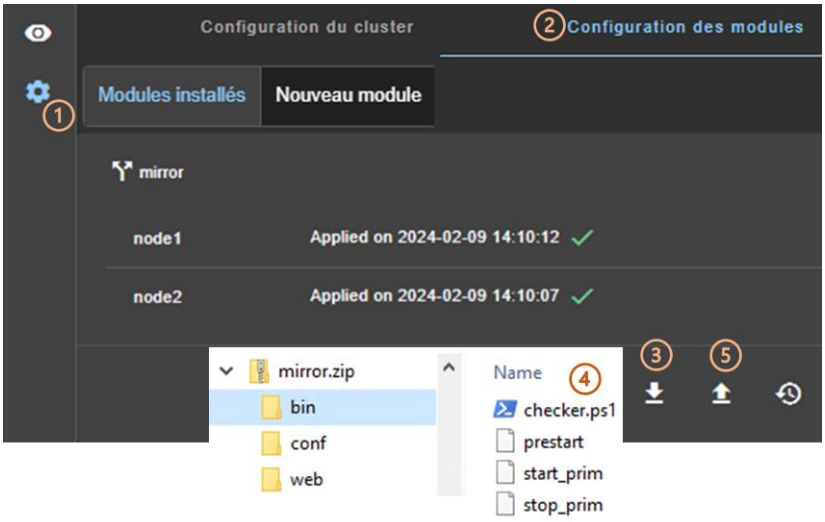
  À la désinstallation du module, l'ensemble de ses fichiers sont empaquetés dans un fichier `.safe` qui est archivé du côté serveur sous `SAFE/Application_Modules/backup`.

- Pour configurer un nouveau module, cliquer sur « Nouveau module ».

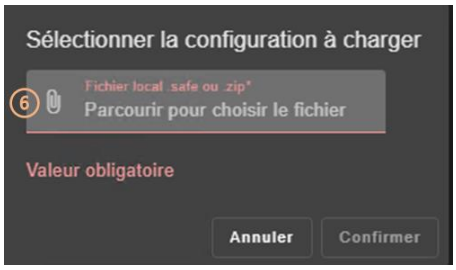
3.3.4 Éditer localement la configuration du module puis l'appliquer

Vous préférerez peut-être utiliser votre éditeur favori sur votre station de travail pour éditer la configuration du module et/ou ajouter des scripts, tels qu'un custom checker.

Suivez la procédure suivante pour éditer la configuration du module sur votre station de travail puis l'appliquer.



- (1) Cliquer sur « Configuration » dans la barre de navigation latérale.
- (2) Cliquer sur l'onglet « Configuration des modules ».
- (3) Cliquer sur pour télécharger le `mirror.safe` sur votre station de travail
- (4) Extraire le contenu de `mirror.safe` (qui est un fichier zip) pour éditer `conf/userconfig.xml`, éditer/ajouter/détruire des scripts sous `bin` (un custom checker par exemple).
- (5) Compresser le répertoire modifié dans un fichier `xx.safe` (ou `xx.zip`), puis le charger avec (.safe et .zip acceptés).



- (6) Cliquer sur pour sélectionner le fichier à charger, puis « Confirmer ».

L'assistant de configuration du module est lancé avec le nouveau contenu de ce fichier. Passez à l'étape 4 pour « Sauvegarder et appliquer » cette nouvelle configuration

3.4 Superviser un module

Une fois un module configuré, vous pouvez superviser son état et exécuter des actions dessus (start, stop…).

La page d'accueil de supervision des modules est accessible :

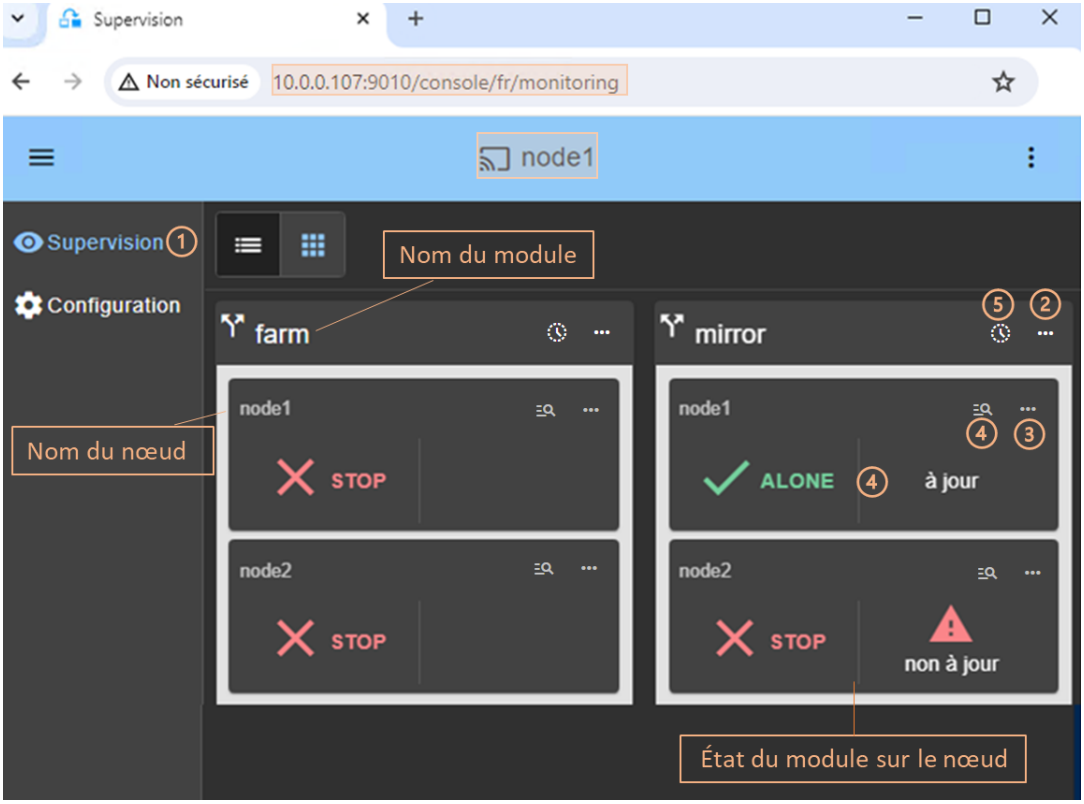
- Directement avec http://host:9010/console/fr/monitoring

Ou

- En naviguant dans la console sur « Supervision »

3.4.1 Page d'accueil de la supervision

Dans cet exemple, la console est chargée à partir de `10.0.0.107`, qui correspond à `node1` dans le cluster. Il s'agit du nœud de connexion. Deux modules sont configurés : `farm` et `mirror`.



- (1) Cliquer sur « Supervision » dans la barre de navigation latérale
- Pour chaque module installé, sont affichés :
  - le nom du module et des nœuds sur lesquels il est installé
  - l'état du module sur chaque nœud (voir section 3.4.2)
  - une notification pour chaque changement d'état, si l'utilisateur a accordé la permission et l'URL est https, ou http://localhost
- (2) Cliquer sur ••• pour ouvrir le menu d'actions (start, stop…) globales sur le module, qui s'appliquent à tous les nœuds (`node1`, `node2` dans l'exemple).

  Pour sa description, voir section 3.4.3.1.
- (3) Cliquer sur ••• pour ouvrir le menu d'actions (start, stop…) locales sur le module, qui s'appliquent uniquement au nœud (`node1` dans l'exemple).

  Pour sa description, voir section 3.4.3.2.
- (4) Cliquer sur le panneau du nœud (`mirror>node1` dans l'exemple) pour ouvrir les détails du module sur ce nœud (journaux, ressources…). Depuis SafeKit 8.2.2, cliquer plutôt sur pour ouvrir/fermer les détails.

  Pour sa description, voir section 3.4.4.
- (5) Cliquer sur pour ouvrir/fermer la chronologie des états du module sur tous les nœuds où il est installé.

  Pour sa description, voir section 3.4.5.

3.4.2 État du module

Le module est représenté par son état synthétique dans la partie gauche et par son état détaillé dans la partie droite.

3.4.2.1 État synthétique

La console affiche l'un des états synthétiques suivants pour le module sur le nœud.

✗ `STOP (NotReady)` (rouge)	Module arrêté (prêt à démarrer)
`WAIT (Transient)` (orange)	État transitoire du module
`ALONE (Transient)` (orange)	État transitoire du module miroir primaire sans secondaire
✓ `ALONE (Ready)` (vert)	État stable du module miroir primaire sans secondaire
`PRIM (Transient)` (orange)	État transitoire du module miroir primaire avec secondaire
✓ `PRIM (Ready)` (vert)	État stable du module miroir primaire avec secondaire

SECOND (Transient) (orange)	État transitoire du module miroir secondaire avec primaire, pendant la phase de resynchronisation des répertoires répliqués
SECOND (Ready) (vert)	État stable du module miroir, secondaire avec primaire
UP (Transient) (orange)	État transitoire du module ferme
UP (Ready) (vert)	État stable du module ferme
WAIT (NotReady) (rouge)	État bloqué du module en attente d'une ou plusieurs ressources
NOT CONFIGURED (gris)	Module installé mais non configuré
ERROR (rouge)	Pas de réponse du nœud dans le délai imparti

Cela peut être dû à une mauvaise adresse, une défaillance du réseau ou du serveur, une mauvaise configuration du navigateur web ou du pare-feu, l'arrêt du service web SafeKit sur le nœud (voir section 7.1).

Cela peut également être dû à l'indisponibilité temporaire du nœud de connexion. Dans ce cas, rechargez la console à partir d'un autre nœud du cluster.

Pour la description des changements d'états d'un module miroir voir la section 5.2.

Pour la description des changements d'états d'un module ferme voir la section 6.2.

3.4.2.2 État détaillé

Il s'agit de l'état des principales ressources et règles de failover.

à jour	Les répertoires répliqués du module miroir sont à jour
non à jour	Les répertoires répliqués du module miroir ne sont pas à jour
degradé	Le module miroir est en mode dégradé décrit en section 7.6
50%, 100%	La part de partage de charge du module ferme (par exemple 50% ou 100% pour 2 nœuds)
0%	Aucune charge prise par le module ferme

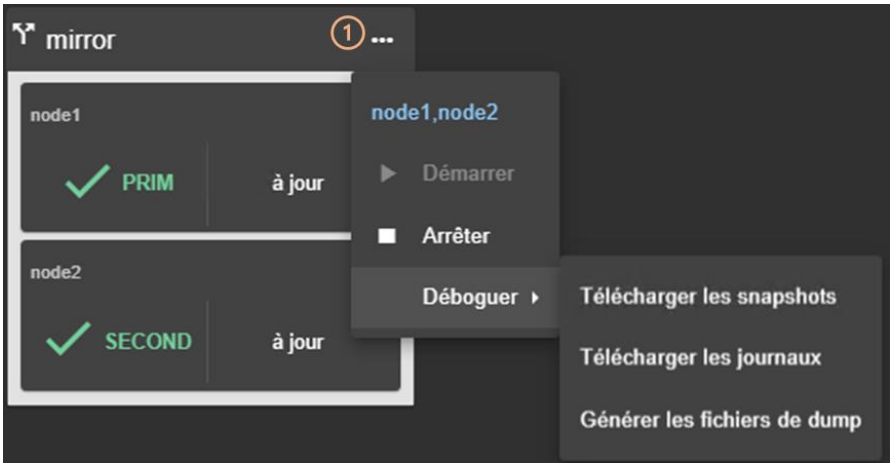
c_checkfile	Le module a appliqué la règle de failover (par exemple, la règle nommée c_checkfile) qui entraîne l'action restart, stop, stopstart ou wait sur le module en raison d'une ressource passé à down. Voir la section 13.18.4.2 pour des détails sur les règles de failover. Pour analyser le problème, lisez les journaux et les états des ressources comme décrit plus loin
HTTP erreur de connexion	Avec le module dans l'état ERREUR (rouge) Pas de réponse du nœud dans le délai imparti

3.4.3 Menus de contrôle d'un module

3.4.3.1 Menu global

Les actions du menu global s'appliquent à tous les nœuds sur lesquels le module est configuré.

Dans cet exemple, les actions s'appliquent au module mirror sur node1 et node2.



- (1) Cliquer sur ••• pour ouvrir le menu d'actions globales.
- Cliquer sur « Démarrer » pour démarrer le module sur tous les nœuds.

  Pour un module miroir, un nœud est démarré automatiquement en tant que primaire s'il a les données à jour ; sinon, il démarre en tant que secondaire.
- Cliquer sur « Arrêter » pour arrêter le module sur tous les nœuds.

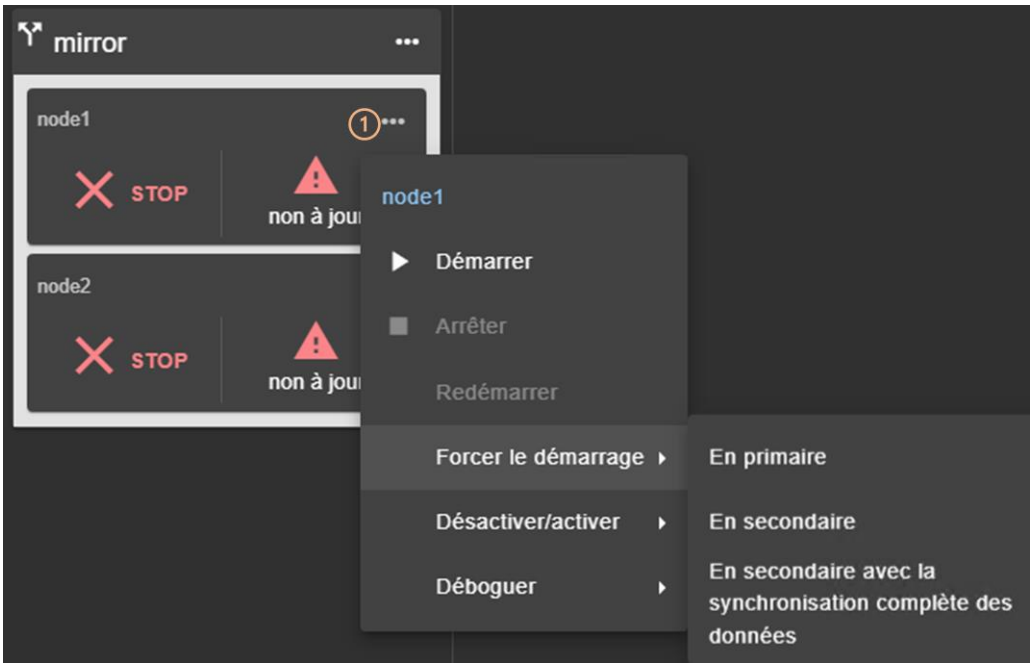
  Pour un module miroir, le nœud qui est secondaire est arrêté en premier pour éviter un basculement inutile.
- Cliquer sur « Déboguer » pour le débogage et support comme décrit en section 3.5.

3.4.3.2 Menu local

Les actions du menu local s'appliquent uniquement au nœud sélectionné.

3.4.3.2.1 Contrôler un module miroir

Dans cet exemple, les actions s'appliquent au module mirror sur node1 uniquement.



- (1) Cliquer sur ••• pour ouvrir le menu d'actions locales sur le nœud désiré (e.g. `node1`).
- Cliquer sur « Démarrer » pour démarrer le module sur le nœud.

  Pour un module miroir, un nœud est démarré automatiquement en tant que primaire s'il a les données à jour ; sinon, il démarre en tant que secondaire. Voir la section 5.5.
- Cliquer sur « Arrêter » pour arrêter le module sur le nœud.
- Cliquer sur « Redémarrer » pour redémarrer le module sur le nœud.

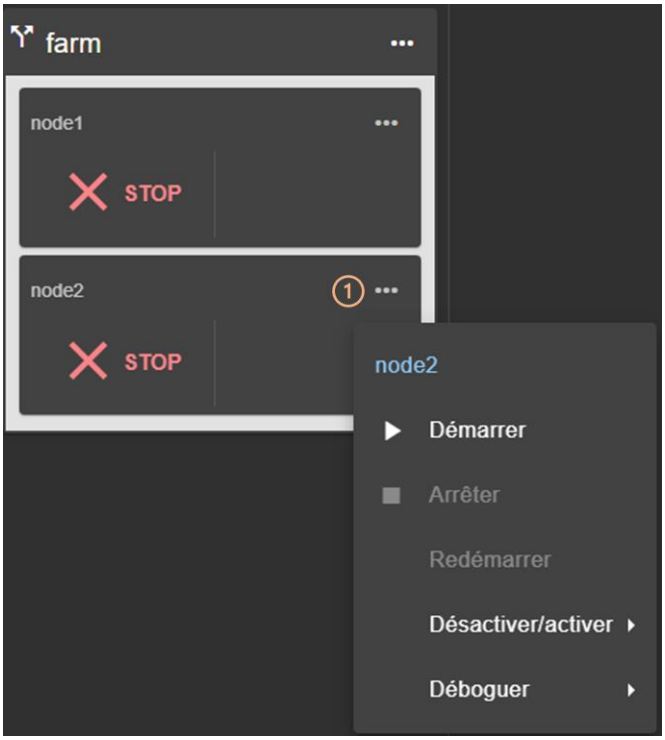
  Il exécute uniquement les scripts d'arrêt puis de démarrage pour redémarrer l'application localement sans entraîner de basculement
- Utiliser le sous-menu « Forcer le démarrage » lorsque vous devez décider quel nœud doit démarrer en primaire ou secondaire.
  - Sélectionner « En primaire » pour forcer le nœud à démarrer en tant que primaire.

    Par exemple, au 1er démarrage d'un module miroir tel que décrit en section 5.3, vous devez « Forcer le démarrage En primaire » le nœud qui a les répertoires répliqués à jour.
  - Sélectionner « En secondaire » pour forcer le nœud à démarrer en tant que secondaire.
  - La synchronisation des données peut être optimisée ou non en fonction du dernier état interne du module.
  - Sélectionner « En secondaire avec la synchronisation complète des données » pour forcer le nœud à démarrer en tant que secondaire après recopie complète des données répliquées.
- Cliquer sur « Désactiver/activer » pour contrôler la détection d'erreur telle que décrite dans la section 3.4.3.2.3.
- Cliquer sur « Déboguer » pour télécharger le journal ou snapshot du module depuis le nœud plutôt que depuis tous les nœuds tels que décrit dans la section 3.5.

Pour comprendre et vérifier le bon fonctionnement d'un module miroir, voir la section 5.
Pour le tester, voir la section 4.

3.4.3.2.2 Contrôler un module ferme

Dans cet exemple, les actions s'appliquent au module `farm` sur `node2` uniquement.



- (1) Cliquer sur ••• pour ouvrir le menu d'actions locales sur le nœud désiré (e.g. `node2`).
- Cliquer sur « Démarrer » pour démarrer le module sur le nœud.
- Cliquer sur « Arrêter » pour arrêter le module sur le nœud.
- Cliquer sur « Redémarrer » pour redémarrer le module sur le nœud.

  Il exécute uniquement les scripts d'arrêt puis de démarrage pour redémarrer l'application localement sans entraîner de basculement
- Cliquer sur « Désactiver/activer » pour contrôler la détection d'erreur telle que décrite dans la section 3.4.3.2.3.
- Cliquer sur « Déboguer » pour télécharger le journal ou snapshot du module depuis le nœud plutôt que depuis tous les nœuds tels que décrit dans la section 3.5.

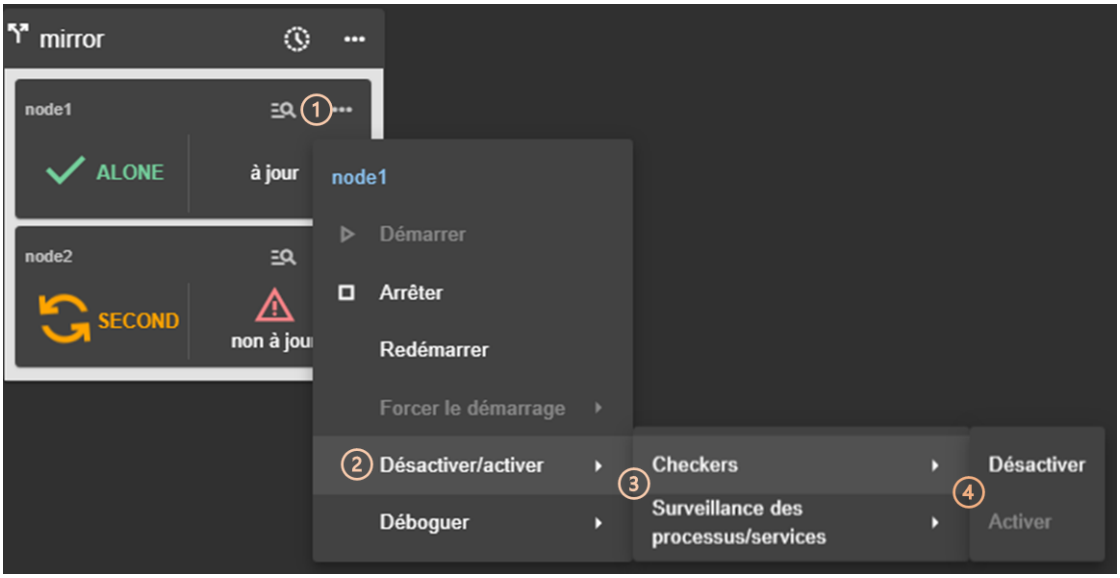
Pour comprendre et vérifier le bon fonctionnement d'un module ferme, voir la section 6.
Pour le tester, voir la section 4.

3.4.3.2.3 Contrôler les checkers ou la surveillance des processus/services

Pour éviter une détection d'erreur infondée et un basculement automatique lors de la maintenance de l'application, vous pouvez désactiver les checkers configurés (TCP, ping, custom, etc.) ou la surveillance des processus/services. Une fois la maintenance terminée, ils peuvent être réactivés en toute sécurité. Ces actions peuvent être

effectuées lorsque le module est démarré/arrêté et ne sont pas réinitialisées lorsque le module redémarre.

Dans l'exemple ci-dessous, les actions s'appliquent au `mirror` sur `node1`.



- (1) Cliquer sur ••• pour ouvrir le menu d'actions locales sur le nœud désiré (e.g. `node1`).
- (2) Cliquer sur « Désactiver/activer » pour ouvrir le sous-menu.
- (3) Cliquer sur « Checkers » ou « Surveillance des processus/services » pour ouvrir le sous-menu.
- (4) Cliquer sur « Désactiver » pour désactiver la détection d'erreur.

  Cela désactive tous les checkers (TCP, ping, custom, etc.) ou la surveillance des processus/services configurés dans le module.
- (4) Cliquer sur « Activer » pour réactiver la détection d'erreur.

  Cela réactive tous les checkers (TCP, ping, custom, etc.) ou la surveillance des processus/services configurés dans le module.

3.4.4 Détails du module

Vous pouvez afficher les détails d'un module sur un nœud :

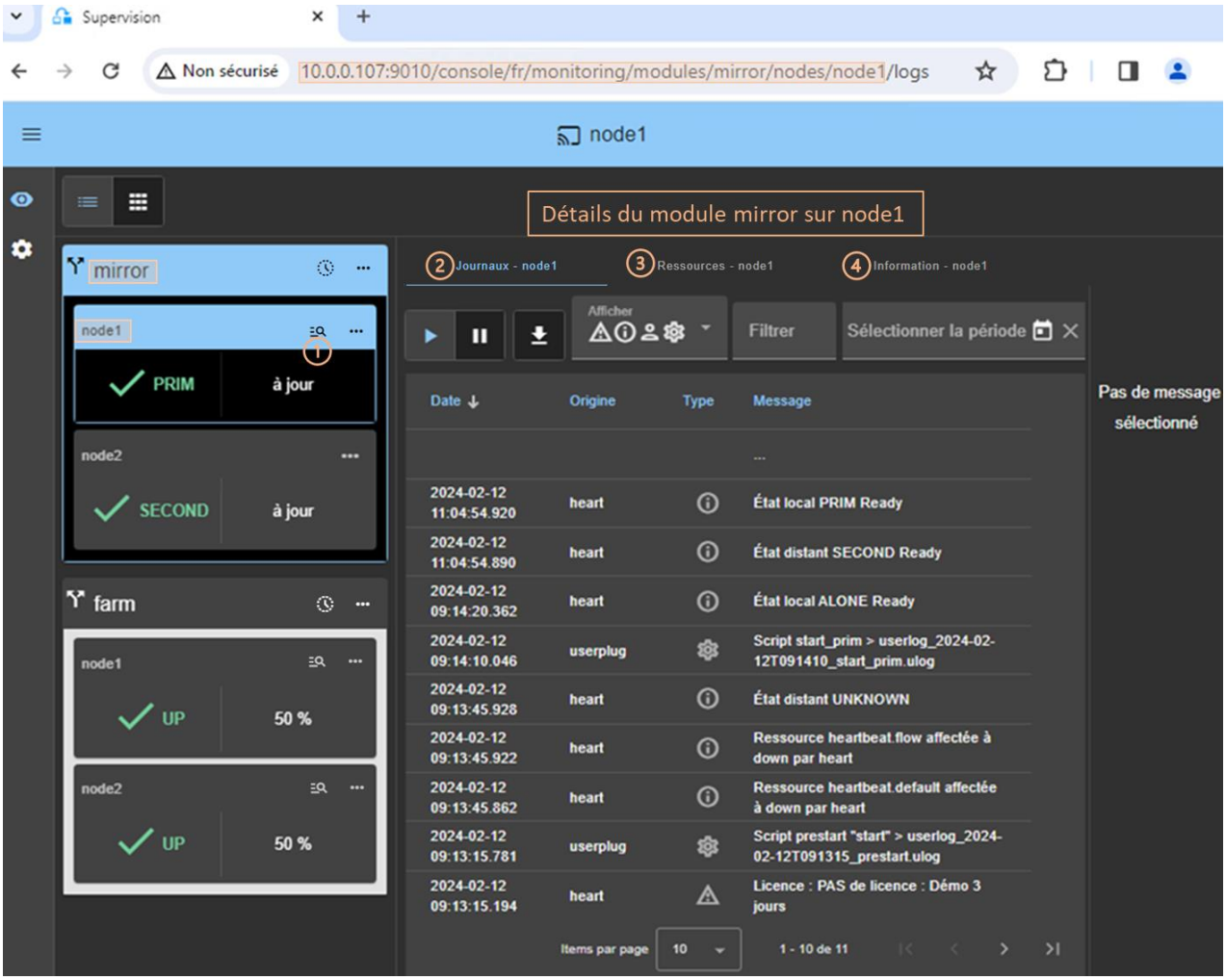
- Directement via l'URL http://host:9010/console/fr/monitoring/modules/*AM*/nodes/*node* (remplacer *AM* par le nom du module et *node* par le nom du nœud)

Ou

- En naviguant dans la console sur « Supervision/Cliquer sur module>nœud »

Le module>nœud sélectionné est mis en évidence par une couleur bleue.

Dans l'exemple suivant, les détails du module `mirror` sur le `node1` sont affichés.



- (1) Cliquer sur pour ouvrir/fermer les détails du module sur ce nœud (logs, ressources...).
- (2) Cliquer sur l'onglet « Journaux » pour visualiser les journaux du module.
- (3) Cliquer sur l'onglet « Ressources » pour visualiser les ressources du module.
- (4) Cliquer sur l'onglet « Information » pour visualiser les informations sur le nœud.

  Il s'agit du nom des lans et adresses définies dans la configuration du cluster, version et licence SafeKit, hostname et OS.

3.4.4.1 Journaux du module

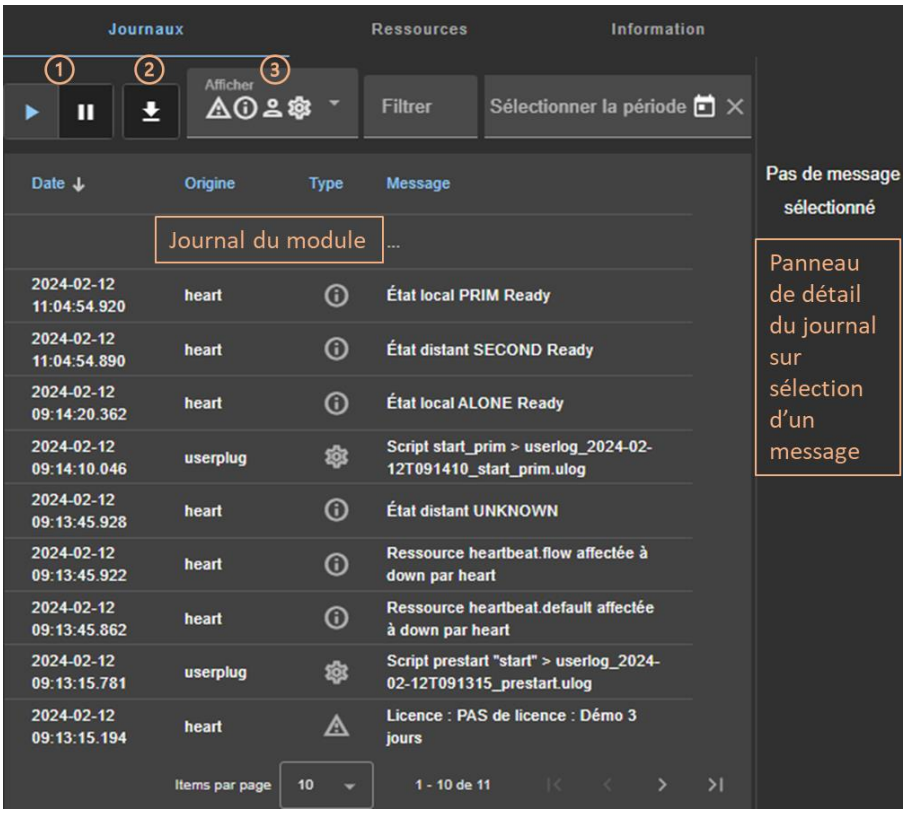
Vous pouvez afficher les journaux d'un module sur un nœud :

- Directement via l'URL http://host:9010/console/fr/monitoring/modules/*AM*/nodes/*node*/logs (remplacer *node* par le nom du nœud et *AM* par le nom du module)

Ou

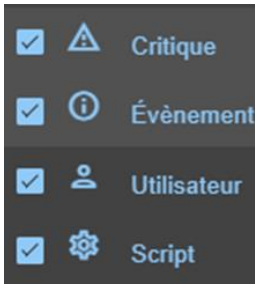
- En naviguant dans la console sur « Supervision/Cliquer sur module>nœud/Onglet Journaux »

Le panneau de gauche affiche en temps réel le journal non verbeux du module>nœud sélectionné.



- Cliquer sur ▶II pour reprendre/suspendre la visualisation en temps réel du journal du module.

  Voir la section 7, pour une explication des principaux messages.
- (2) Cliquer sur ⬇ pour télécharger le journal du module (non verbeux ou verbeux).
- (3) Sélectionner le type de messages à afficher :

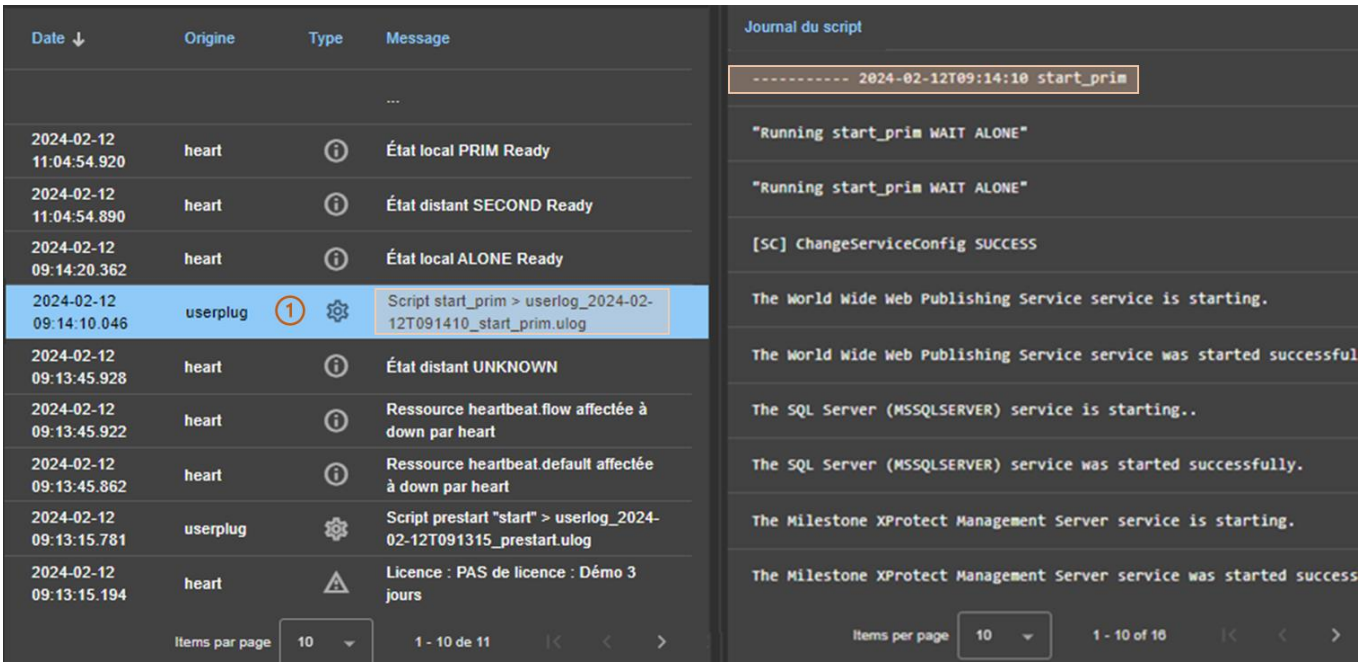


- C(ritique) messages tels que des détections d'erreurs
- E(vènement) messages tels que l'état local et distant
- U(ser) messages lorsque l'utilisateur exécute une action sur le module
- S(cript) messages quand les scripts du module sont exécutés

- Cliquer sur un message pour afficher le journal verbeux du module ou le journal des scripts (sortie des scripts) dans le détail du journal dans le panneau de droite.

3.4.4.1.1 Journal du script

Pour afficher le journal du script du module, cliquer sur le message ⚙S(cript) dont vous souhaitez visualiser l'output.

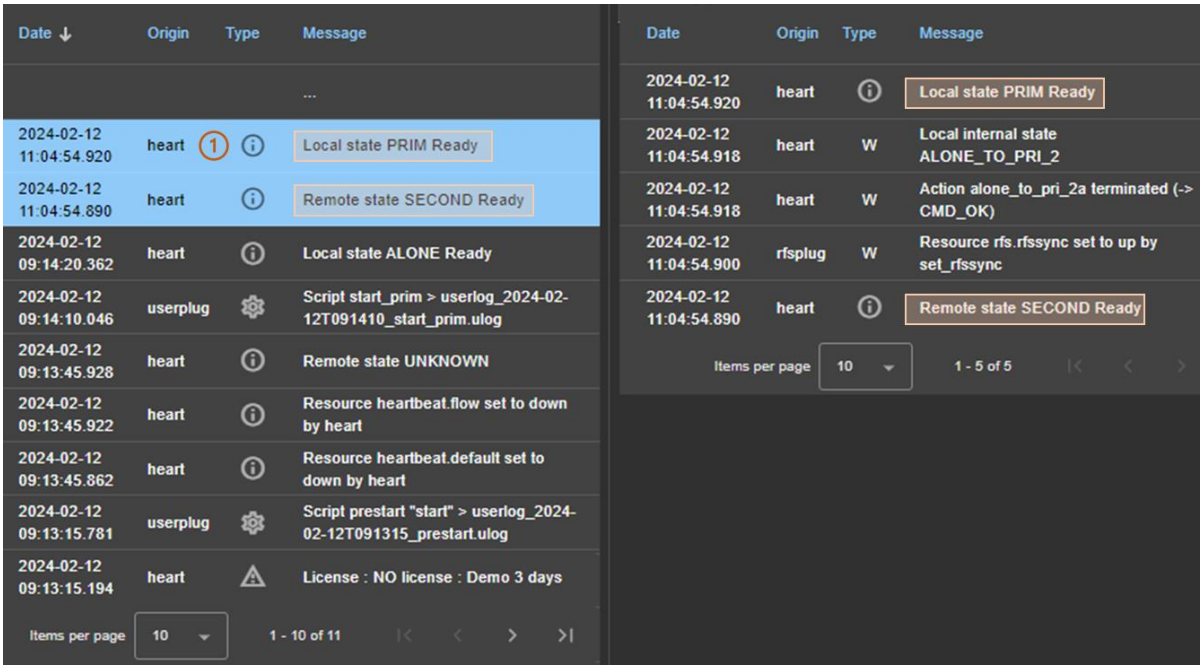


- (1) Cliquer sur le message ⚙S(cript) qui consiste en :
  - la date et l'heure d'exécution du script
  - le nom du script exécuté
  - le nom du fichier `userlog` correspond

Le contenu du fichier `userlog` est affiché dans le panneau de droite. Dans l'exemple, il s'agit du contenu du fichier `SAFEVAR/modules/`*`AM`*`/userlog_2024-02-12T091410_start_prim.ulog` (où *AM* est le nom du module).

3.4.4.1.2 Journal verveux

Pour afficher le journal verbeux du module, cliquer sur un message autre que ⚙S(cript).



- (1) Cliquer sur le message qui consiste en :
  - la date et l'heure de l'évènement
  - le message du module
- Tous les messages verbeux entre le message sélectionné et le précédent dans la table, sont affichés dans le panneau de droite.

3.4.4.2 Ressources du module

Vous pouvez afficher les ressources d'un module sur un nœud :

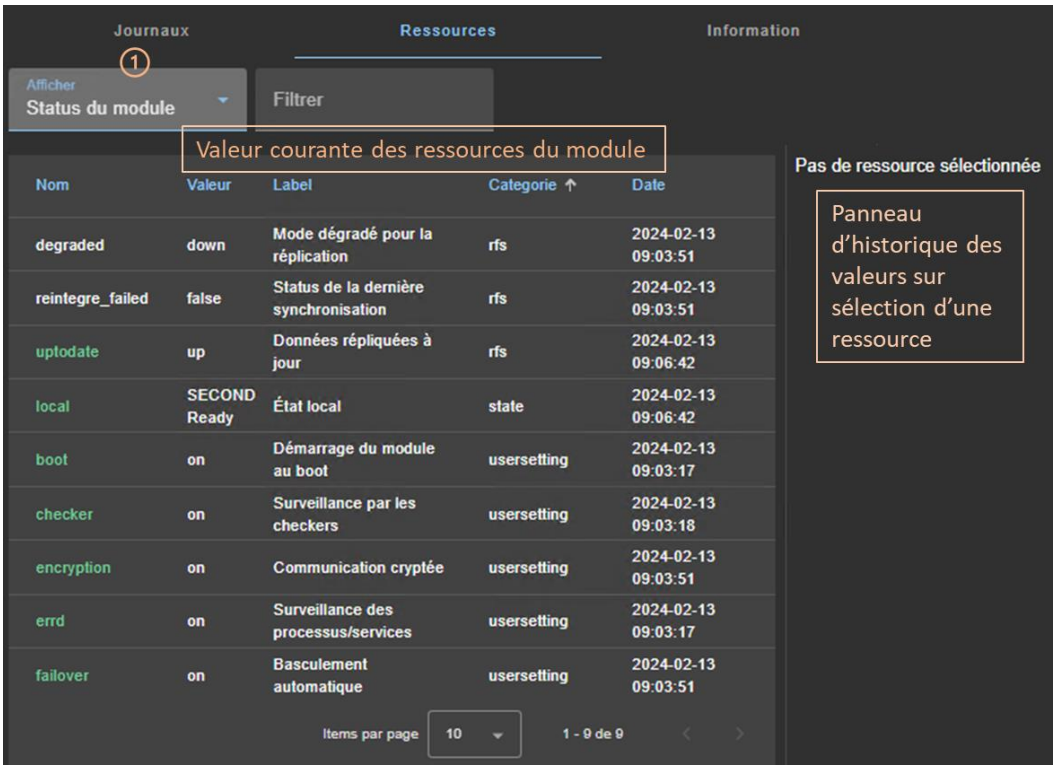
- Directement via l'URL http://host:9010/console/fr/monitoring/modules/*AM*/nodes/*node*/resources (remplacer *AM* par le nom du module et *node* par le nom du nœud)

Ou

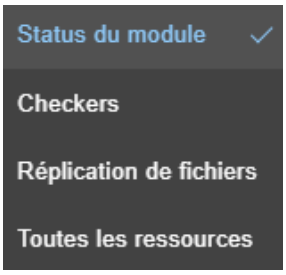
- En naviguant dans la console sur « Supervision/Cliquer sur module>nœud/Onglet Ressources »

3.4.4.2.1 État courant des ressources

Le panneau de gauche affiche en temps réel l'état courant des ressources du module>nœud sélectionné.



- (1) Sélectionner le groupe de ressources à afficher



- Status du module

  Ressources principales, notamment de la réplication de fichiers pour un module miroir
- Checkers

  Ressources affectées par des checkers
- Réplication de fichiers

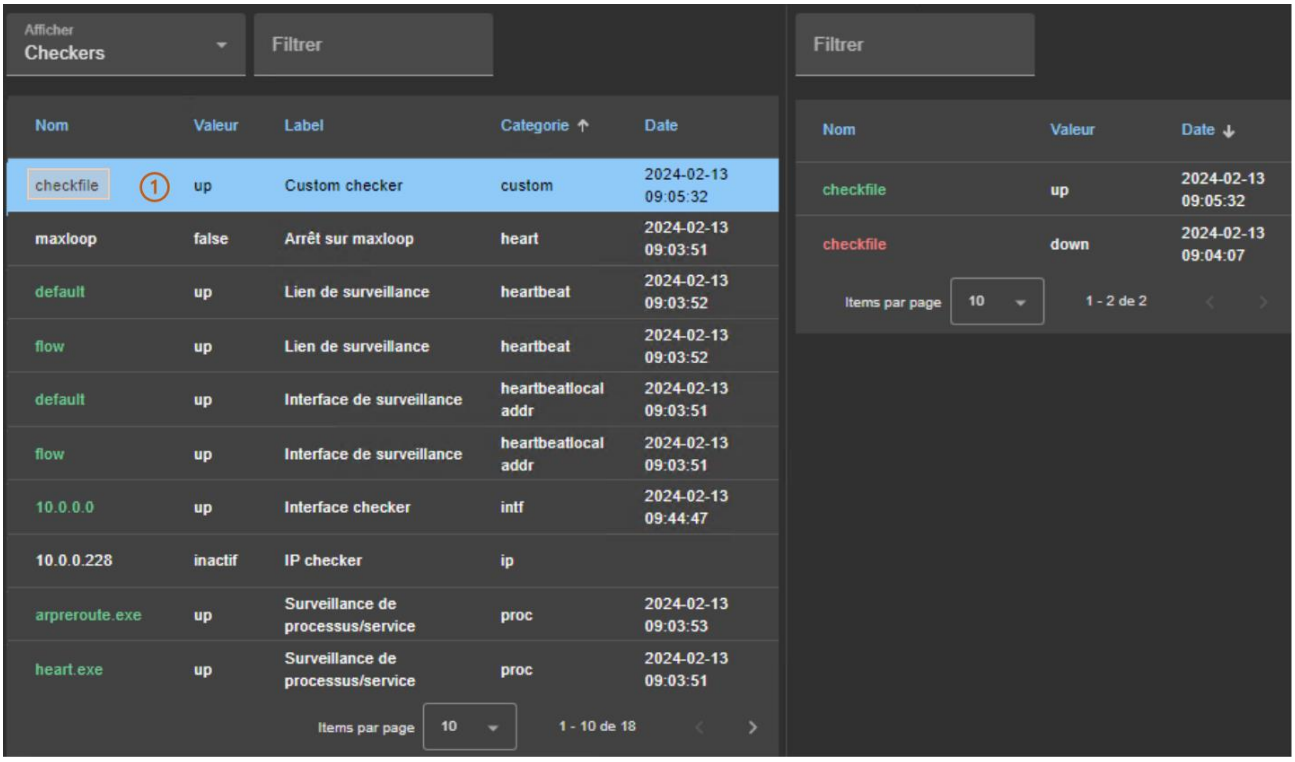
  Ressources spécifiques à la réplication de fichiers qui démontrent la progression de la synchronisation
- Toutes les ressources

- Cliquer sur une ressource pour afficher la valeur de la ressource dans le temps dans le panneau de droite. Cet historique peut être vide pour certaines ressources (non affecté ou nettoyé).

L'état courant des ressources du module est contrôlé par la failover machine pour provoquer des actions sur le module en cas de défaillance (voir section 13.18).

3.4.4.2.2 Historique des valeurs d'une ressource

Pour afficher l'historique des valeurs d'une ressource, cliquer sur la ressource qui vous intéresse.



- (1) Cliquer sur la ligne qui consiste en :
  - la dernière date à laquelle la ressource a été affectée
  - le nom et la catégorie de la ressource. Le nom complet de la ressource est de la forme *catégorie*.*nom* (`custom.checkfile` dans l'exemple).

L'historique des valeurs de la ressource est affiché dans le panneau de droite. Dans l'exemple, il s'agit de la ressource `custom.checkfile` correspondant à une ressource affectée par un custom checker.

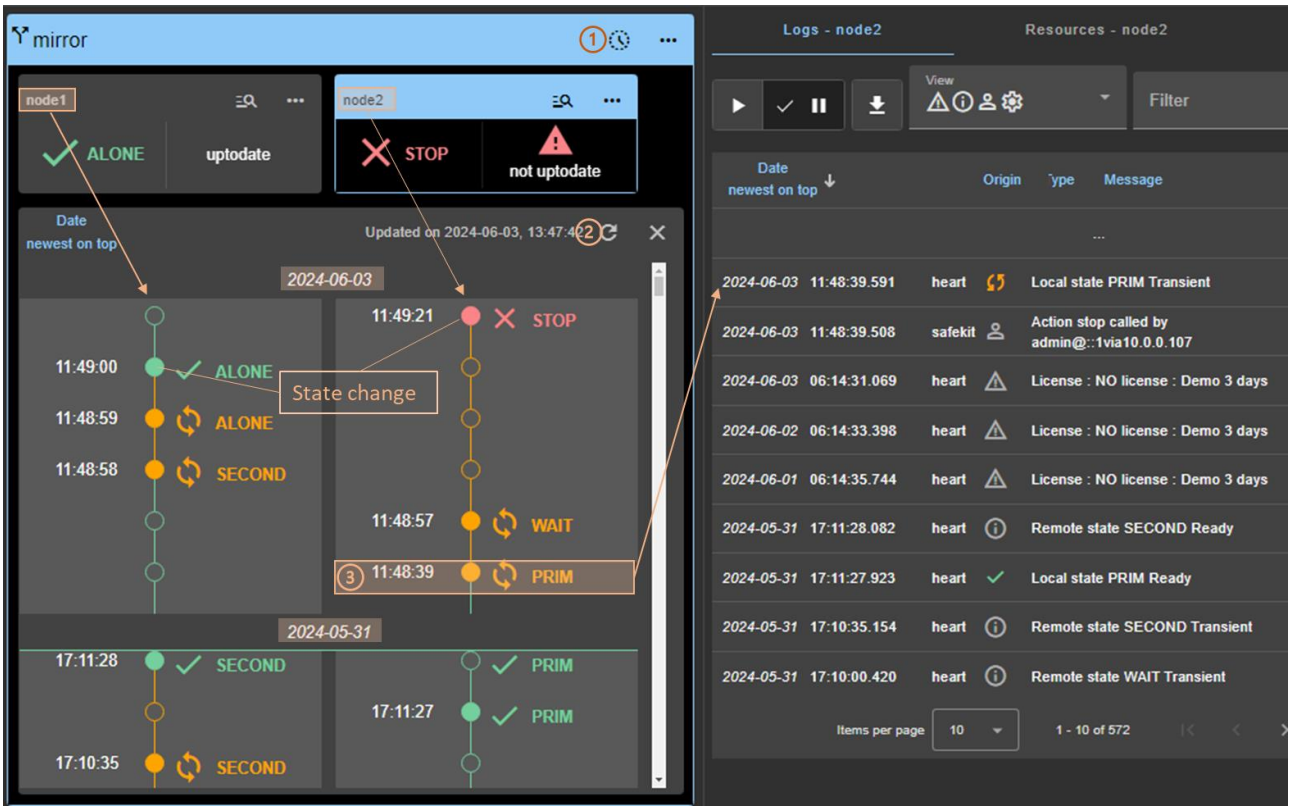
3.4.5 Chronologie des états du module

Depuis SafeKit 8.2.2, vous pouvez afficher la chronologie des états d'un module :

- En naviguant dans la console sur « Supervision/Cliquer sur pour le module »

Cela permet d'avoir une vue globale de l'état du module sur le cluster. Notez que :

- les horloges des deux nœuds doivent être synchronisées pour que la mise en correspondance des changements d'état soit significative.
- la chronologie affichée est inversée : les états du module sur tous les nœuds au fil du temps, en commençant par la date la plus récente.



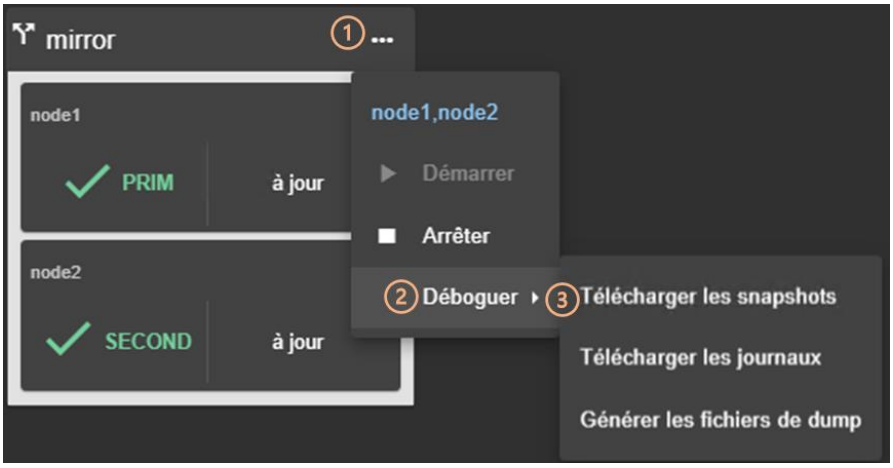
- (1) Cliquer sur pour ouvrir/fermer la chronologie. La chronologie affichée est celle disponible au moment du chargement.
- (2) Cliquer sur pour rafraîchir la chronologie avec les derniers changements d'état.
- (3) Cliquer sur un événement de changement d'état pour afficher le journal du module pour le nœud à partir de cette date.

3.5 Snapshots et journaux du module pour le débogage et support

Lorsque le problème n'est pas facilement identifiable, il est recommandé de prendre un snapshot du module sur tous les nœuds, comme décrit ci-dessous. Les snapshots

permettent une analyse hors ligne et approfondie de l'état du module et du nœud, tel que décrit dans la section 7.16. Si cette analyse échoue, envoyez les snapshots au support comme décrit dans la section 8.

Dans l'exemple suivant, le module `mirror` est configuré sur `node1` et `node2`. Notez qu'un snapshot peut être téléchargé dans n'importe quel état du module.



- (1) Cliquer sur ••• pour ouvrir le menu global sur le module.
- (2) Cliquer sur « Déboguer » pour ouvrir le sous-menu de débogage.
- (3) Cliquer sur « Télécharger les snapshots » pour créer et télécharger le snapshot du module pour chaque nœud.

  La console web s'appuie sur les paramètres de téléchargement du navigateur web pour sauvegarder les snapshots sur la station de travail. Certains navigateurs peuvent demander une confirmation pour télécharger plusieurs fichiers et des .zip.

  La commande de génération du snapshot génère un nouveau dump et créé un fichier .zip qui contient les 3 derniers dumps et les 3 dernières configurations du module.

  Dans cet exemple, 2 snapshots sont téléchargés : `snapshot_node1_mirror.zip` et `snapshot_node2_mirror.zip`.
- Cliquer sur « Télécharger les journaux » pour télécharger le journal du module (verbeux ou non) pour chaque nœud.
- En cas de problèmes de réplication de fichiers, il peut être nécessaire de « Générer les fichiers de dump » au moment où le problème se produit.

  Le dump contient les journaux du module et des informations sur l'état du système et de SafeKit au moment du dump. Il est généré du côté serveur sous `SAFEVAR/snapshot/modules/mirror/dump_AAAA_MM_DD_hh_mm_ss`.

3.6 Sécuriser la console web

SafeKit propose différentes politiques de sécurité pour la console web mises en œuvre en modifiant la configuration du service web SafeKit. Ces configurations offrent aussi une gestion de rôles :

Rôle Admin ⚙👁	Ce rôle accorde tous les droits d'administration en autorisant l'accès à la ⚙ Configuration et 👁 Supervision dans la barre de navigation latérale

Rôle Control 	Ce rôle accorde tous les droits de supervision et contrôle en autorisant l'accès à la  Supervision
Rôle Monitor 	Ce rôle n'accorde que des droits de supervision, interdisant les actions sur les modules (démarrage, arrêt...) sous  Supervision

SafeKit fournit différentes configurations pour le service web afin de renforcer la sécurité de la console. Les configurations prédéfinies sont listées ci-dessous de la moins sécurisée à la plus sécurisée :

- HTTP. Rôle identique pour tous les utilisateurs sans authentification

  Cette solution ne peut être mise en œuvre qu'en HTTP et est incompatible avec les méthodes d'authentification des utilisateurs.

- HTTP/HTTPS avec authentification à base de fichiers et gestion de rôles facultative

  Elle repose sur le fichier Apache `user.conf` pour authentifier les utilisateurs et, optionnellement, restreindre leurs accès en fonction des rôles avec le fichier `group.conf`. La connexion à la console nécessite la saisie du nom et du mot de passe de l'utilisateur tels qu'ils ont été configurés avec les mécanismes d'Apache.

  Il s'agit de la configuration par défaut, en HTTP et initialisée avec un seul utilisateur `admin` ayant le rôle Admin. Cette configuration peut être étendue pour rajouter des utilisateurs ou passer en HTTPS.

- HTTP/HTTPS avec authentification à base de serveur LDAP/AD. Gestion de rôles facultative

  Elle repose sur le serveur LDAP/AD pour authentifier les utilisateurs et, optionnellement, restreindre leurs accès en fonction des rôles. La connexion à la console nécessite la saisie de l'identifiant et du mot de passe de l'utilisateur tels qu'ils ont été configurés dans le serveur LDAP/AD. Elle peut être appliquée en HTTP ou HTTPS.

- HTTPS avec authentification à base de serveur d'OpenId Connect. Gestion de rôles facultative

  Elle repose sur le serveur OpenID Identity Provider pour authentifier les utilisateurs et, optionnellement, restreindre leurs accès en fonction des rôles. La connexion à la console nécessite la saisie de l'identifiant et du mot de passe de l'utilisateur tels qu'ils ont été configurés dans le serveur OpenID. Depuis SafeKit 8.2.3, l'authentification OpenId ne peut être utilisée qu'avec HTTPS.

Pour les mettre en œuvre, se référer à la section 11.

4. Tests

Les tests suivants permettent de mieux comprendre le fonctionnement de SafeKit et de s'assurer que la solution déployée retourne bien les résultats attendus. Ils peuvent être utilisés comme base de cahier de recette chez un client.

Dans la suite, l'analyse des résultats des tests peut nécessiter de consulter le journal du module, le journal des scripts (qui contient l'output des scripts du modules) ou l'état des ressources du module. Pour cela, voir la section 7.3.

4.1 Installation et tests après boot

4.1.1 Test installation package

Dans la suite, remplacer `node1` par le nom du nœud et *AM* par le nom du module.

- `safekit -p` executé sur un nœud retourne, entre autres valeurs, la valeur de `SAFE`, le chemin d'installation racine de SafeKit, et `SAFEVAR`, le répertoire de travail de SafeKit :
  - en Windows

    ```
    SAFE=C:\safekit si SystemDrive=C:
    SAFEVAR=C:\safekit\var
    ```

  - en Linux

    ```
    SAFE="/opt/safekit"

    SAFEVAR="/var/safekit"
    ```

  Pour plus d'informations, voir la section 10.1.
- L'édition de **`userconfig.xml`** d'un module miroir (/ferme) et de ses scripts **`start_prim/start_both`**, **`stop_prim/stop_both`** est réalisée avec :
- La console web avec l'URI /console/fr/configuration/modules/*AM*/config
- Sous le répertoire `SAFE/modules/`*AM* sur `node1`
- Le journal du module et le journal des scripts (qui contient output des scripts du module) peuvent être consultés avec :
  - la console web avec l'URI /console/fr/monitoring/modules/*AM*/nodes/node1/logs
  - la commande `safekit logview -m` *AM* exécutée sur `node1`, pour le journal du module
  - sur node1, dans les fichiers `SAFEVAR/modules/`*AM*`/userlog_<year>_<month>_<day>T<time>_<script name>.ulog`, pour le journal des scripts

4.1.2 Test licence et version

`safekit level` retourne

```
Host : <hostname>
OS : <OS version>
SafeKit : <SafeKit version>
License : Demo (No license)| Invalid Product | Invalid Host | … Expiration… | <license id> for <hostname>…
or License : Expired license
```

- "No license"

  signifie qu'il n'y a pas de fichier `SAFE/conf/license.txt` : le produit s'arrête tous les 3 jours
- "Invalid Product"

  signifie que la licence a expiré dans `SAFE/conf/license.txt`
- "Invalid Host"

  signifie que le hostname est invalide dans `SAFE/conf/license.txt`
- " …Expiration…"

  indique une clé temporaire
- "<license id> for <hostname>"

  indique une clé permanente

`http://www.evidian.com/safekit/requestevalkey.php` pour obtenir une clé temporaire d'un mois pour n'importe quel hostname/OS.

`https://support.evidian.com` pour obtenir une clé permanente basée sur le hostname et l'OS.

4.1.3 Test des services et modules SafeKit après boot

Pour Windows, voir aussi la section 10.4.

Test du service `safeadmin`

Le service `safeadmin` doit être démarré automatiquement au boot. Pour vérifier son état :

En Windows	1. Ouvrir une console PowerShell en tant qu'administrateur 2. Exécuter `Get-Service -name safeadmin` `Status Name DisplayName` `------ ---- -----------` `Running safeadmin safeadmin`
En Linux	1. Ouvrir une console Shell en tant que root 2. Exécuter `systemctl status safeadmin` `Redirecting to /bin/systemctl status safeadmin.service` `● safeadmin.service - The SafeKit Administration Daemon` `Loaded: loaded (/usr/lib/systemd/system/safeadmin.service; enabled; vendor preset: disabled)`

	Active: active (running) since Tue 2024-11-12 17:30:56 CET; 20h ago …

Lorsque le service `safeadmin` n'est pas actif, toutes les commandes `safekit` échouent et retournent par exemple :

```
safekit level
```

```
Waiting for safeadmin .....
Error: safeadmin administrator daemon not running
```

Voir la section 9.1.1 pour démarrer le service `safeadmin`.

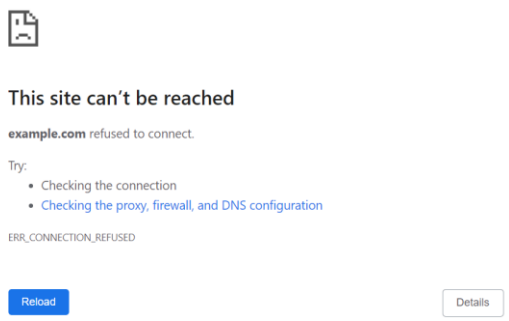
Test du service `safewebserver`

Par défaut, le service `safewebserver` est démarré automatiquement au boot. Pour vérifier son état :

En Windows	1. Ouvrir une console PowerShell en tant qu'administrateur 2. Exécuter Run `Get-Service -name safewebserver` Status Name DisplayName ------ ---- ----------- Running safewebserver safewebserver
En Linux	1. Ouvrir une console Shell en tant que root 2. Exécuter `systemctl status safewebserver` systemctl status safewebserver Redirecting to /bin/systemctl status safewebserver.service ● safewebserver.service - SafeKit Apache Server Loaded: loaded (/usr/lib/systemd/system/safewebserver.service; enabled; vendor preset: disabled) Active: active (running) since Wed 2024-11-13 11:01:31 CET; 2h 58min ago …

Lorsque le service `safewebserver` n'est pas actif, les fonctionnalités suivantes ne sont pas opérationnelles :

- La console web SafeKit qui affiche :



- Le module checker

- La commande distribuée qui retourne par exemple :

  `safekit -H "*" level`

```
---------- Server=https://10.0.0.107:9453 ----------
curl: (7) Failed to connect to 10.0.0.107 port 9453 after 1022 ms: Couldn't connect to server
---------- Server=https://10.0.0.108:9453 ----------
curl: (28) Failed to connect to 10.0.0.108 port 9453 after 21024 ms: Couldn't connect to server
```

Voir la section 9.1.2 pour démarrer le service `safewebserver`.

Test du service SNMP

Par défaut, la surveillance SNMP n'est pas activée. Voir la section 10.9 pour l'activer.

En Windows, elle est fournie par le service `Net-SNMP Agent`. En Linux, elle repose sur le service standard `snmpd`. Pour vérifier son état :

En Windows	1. Ouvrir une console PowerShell en tant qu'administrateur 2. Exécuter Run `Get-Service -name "Net-SNMP Agent"` `Status Name DisplayName` `------ ---- -----------` `Running Net-SNMP Agent Net-SNMP Agent`
En Linux	1. Ouvrir une console Shell en tant que root 2. Exécuter `systemctl status snmpd` `systemctl status snmpd` `Redirecting to /bin/systemctl status snmpd.service` `● snmpd.service - …` `Active: active (running) since Wed 2024-11-13 11:01:31 CET; 2h 58min ago` `…`

Lorsque le service n'est pas actif, la surveillance SNMP n'est pas opérationnelle.

Voir la section 9.1.3 pour démarrer le service.

Test des modules

- `safekit boot status` retourne le démarrage ("on") ou non ("off") des modules au boot
- `safekit state` retourne l'état de tous les modules configurés : `STOP` (miroir ou ferme), `WAIT` (miroir ou ferme), `ALONE` (miroir), `PRIM` (miroir), `SECOND` (miroir), `UP` (ferme)
- vérifier les processus d'un module (voir section 10.2)

  Pour lister les processus du module *`AM`*, exécuter :

  `safekit -r processtree list all AM`

  Cette commande retourne tous les processus ayant *`AM`* en argument.
- `safekit module listid` retourne le nom des modules installés et leurs ids : l'id d'un module doit être le même sur tous les serveurs

4.1.4 Test démarrage de la console web

- Connecter un navigateur web à `http://<server IP>:9010`
- La page d'accueil de la console web apparaît

4.2 Tests d'un module miroir

4.2.1 Test du premier start d'un module miroir sur 2 serveurs ✗STOP (NotReady)

Au premier démarrage du module après sa configuration :

- Message dans les journaux du module sur les 2 serveurs (pour lire les journaux, voir la section 7.3)

  "Action start called by admin@<IP>/SYSTEM/root"

- Le module va dans l'état ○ `WAIT (NotReady)` and ○ `WAIT (NotReady)` sur les 2 serveurs et le journal contient les messages

  "Action wait from failover rule rfs_notuptodate_server"
  "Data may be not uptodate for replicated directories (wait for the start of the remote server)"
  "If you are sure that this server has valid data, run safekit stop, then safekit prim to force start as primary"

Pour le premier démarrage d'un module miroir avec de la réplication de fichiers, l'utilisateur doit forcer le démarrage en primaire du serveur ayant les données répliquées à jour. Voir la section 5.3.

4.2.2 Test start d'un module miroir sur 2 serveurs ✗STOP (NotReady)

Pour les démarrages suivants :

- Message dans les journaux du module sur les 2 serveurs (pour lire les journaux, voir la section 7.3)

  "Action start called by admin@<IP>/SYSTEM/root"

- Le module va dans l'état stable ✓`PRIM (Ready)` et ✓`SECOND (Ready)` sur les 2 nœuds avec dans le 1er journal

  "Remote state SECOND Ready"
  "Local state PRIM Ready"

- Et dans le 2nd journal

  "Local state SECOND Ready"
  "Remote state PRIM Ready"

- L'application est démarrée dans le script `start_prim` du module `PRIM` avec le message dans son log

  "Script start_prim"

4.2.3 Test stop d'un module miroir sur le serveur ✓PRIM (Ready)

Sur le serveur qui s'arrête :

- Message dans le journal du module (pour lire les journaux voir la section 7.3)

  "Action stop called by admin@<IP>/SYSTEM/root"

- Le module qui s'arrête exécute le script `stop_prim` pour arrêter l'application sur le serveur avec le message dans son journal :

  "Script stop_prim"

- Le module arrêté devient ✗`STOP (NotReady)` avec les messages dans son journal :

  "Local state STOP NotReady"

Sur l'autre serveur :

- Le module exécute un basculement avec le message dans son journal :

  "Action alone called by heart : remote stop"

- L'application est démarrée avec le script `start_prim` script du module :

  "Script start_prim"

- Le serveur de reprise devient ✓`ALONE (Ready)` avec le message dans son journal :

  "Local state ALONE Ready "

4.2.4 Test start du module miroir dans l'état ✗STOP (NotReady)

Démarrage du module alors que le module est dans l'état ✓ `ALONE (Ready)` sur l'autre serveur :

- Message dans le journal du module redémarré (pour lire les journaux voir la section 7.3)

  "Action start called by admin@<IP>/SYSTEM/root"

- Le module ✗`STOP (NotReady)` devient ✓`SECOND (Ready)`
- Le module sur l'autre serveur passe de ✓`ALONE (Ready)` à ✓`PRIM (Ready)` et continue à exécuter l'application

4.2.5 Test restart du module miroir dans l'état ✓`PRIM (Ready)`

- Message dans le journal du module redémarré (pour lire les journaux voir la section 7.3)

  "Action restart called by admin@<IP>/SYSTEM/root"

- Le module `PRIM` devient ↻`PRIM` (magenta) puis ✓`PRIM (Ready)`
- Les scripts du module `stop_prim`/`start_prim` sont exécutés sur le module `PRIM` et redémarre localement l'application avec les messages dans son journal :

  "Script stop_prim"
  "Script start_prim"

- L'autre serveur reste ✓`SECOND (Ready)`

4.2.6 Test adresse IP virtuelle d'un module miroir

Module miroir dans l'état ✓`PRIM (Ready)` sur le serveur node1 et ✓`SECOND (Ready)` sur le serveur node2.

`userconfig.xml` :

```
<vip>
 <interface_list>
```

1. Sur le serveur node1, `ipconfig /all` (Windows) ou `ip addr show` (Linux) retourne `virtip` en alias sur l'interface réseau.

   Sur la station externe (ou le serveur), les 3 pings répondent

```
 <interface check="on">
 <real_interface>
   <virtual_addr addr="virtip"
              where="one_side_alias"
             check="on"/>
</real_interface>
  </interface>
 </interface_list>
</vip>
```

1. Sur une station de travail externe (ou un serveur) dans le même LAN, ping des 2 adresses IP physiques + adresse IP virtuelle :

   ```
   ping adresse_ip_node2
   ping adresse_ip_node1
   ping virtip
   arp -a
   ```

2. `safekit stopstart -m AM` (où *AM* est le nom du module) sur le serveur primaire
3. Sur la station de travail externe (ou le serveur) dans le même LAN,

   ```
   ping adresse_ip_node2
   ping adresse_ip_node1
   ping virtip
   arp -a
   ```

   Note: refaire le ping vers `virtip` avant de regarder la table ARP car l'entrée peut être marquée obsolète et est rafraichie après le ping

   Sur la station externe (ou le serveur) dans le même LAN, `virtip` est mappé sur l'adresse MAC de node1

   ```
   arp -a
   adresse_ip_node1      00-0c-29-0a-5c-fc
   adresse_ip_node2      00-0c-29-26-44-93
   virtip     00-0c-29-0a-5c-fc
   ```

2. Après le `stopstart` avec ✓`SECOND (Ready)` sur serveur node1 et ✓`PRIM (Ready)` sur serveur node2

   Dans le journal verbeux du nouveau `PRIM`, message :

   ```
   "Virtual IP <virtip of mirror> set"
   ```

3. Sur le serveur node2, `ipconfig /all` (Windows) ou `ip addr show` (Linux) retourne `virtip` en tant qu'alias sur l'interface réseau

   Sur la station externe (ou le serveur), les 3 pings répondent

   Sur la station externe (ou le serveur), `virtip` est mappé sur l'adresse MAC de ip1.2

   ```
   arp -a
   adresse_ip_node1      00-0c-29-0a-5c-fc
   adresse_ip_node2      00-0c-29-26-44-93
   virtip     00-0c-29-26-44-93
   ```

4.2.7 Test réplication de fichiers d'un module miroir

Module miroir dans l'état ✓`PRIM (Ready)` sur serveur node1 et ✓`SECOND (Ready)` sur serveur node2.

`userconfig.xml` en Windows :

```
<rfs>
  <replicated dir="c:\replicated"
mode="read_only" />
</rfs>
```

`userconfig.xml` en Linux :

```
<rfs>
  <replicated dir="/replicated"
mode="read_only" />
</rfs>
```

En Windows, le répertoire répliqué est `c:\replicated` ; en Linux, `/replicated`.

1. Le fichier `file1.txt` a été créé sur le serveur ✓`SECOND (Ready)` dans le répertoire répliqué
2. Échec car le répertoire répliqué `/replicated` est en lecture seule sur le serveur ✓`SECOND (Ready)`
3. Le fichier `file2.txt` n'est pas répliqué dans le répertoire sur le serveur ✗`STOP (NotReady)`
4. Le fichier `file2.txt` est réintégré sur le serveur redémarré. Pendant la phase de réintégration, le serveur est 🔄`SECOND (Transient)`

   Dans le journal du serveur Linux réintégré, message

1. Sur le serveur ✓PRIM (Ready), aller dans le répertoire répliqué et créer 1 fichier file1.txt 2. Sur le serveur ✓SECOND (Ready), aller dans le répertoire répliqué et essayer de détruire file1.txt 3. Arrêter le serveur ✓PRIM (Ready) et attendre qu'il soit ✗STOP (NotReady). Puis aller sur l'autre serveur devenu ✓ALONE (Ready) et créer un nouveau fichier file2.txt 4. Redémarrer le serveur ✗STOP (NotReady) et attendre qu'il soit ✓SECOND (Ready).	"Updating directory tree from /replicated_For_SafeKit_Replication " Dans le journal du serveur Windows réintégré, message "Updating directory tree from c:\replicated" Et à la fin de la réintégration du répertoire, lorsqu'au moins 1 fichier avec des données modifiées a été réintégré de la machine primaire vers la machine secondaire, message "Copied <reintegration statistics>" "Reintegration ended (synchronize)" Ce message donne les statistiques de réintégration du répertoire : taille réintégrée, nombre de fichiers, temps, débit sur le réseau de réplication en KB/sec. Note : réintégrer un fichier de plus de 100 MB pour avoir des statistiques fiables A la fin de la réintégration, le serveur est ✓SECOND (Ready)

4.2.8 Test shutdown du serveur ✓PRIM (Ready)

- Sur Windows, vérifier que la procédure spéciale d'arrêt des modules au shutdown a été réalisée (voir section 10.4)
- Effectuer un shutdown du serveur ✓PRIM (Ready)
- Vérifier dans le journal du serveur ✓SECOND (Ready) le message

  "Action alone called by heart : remote stop"
- Le serveur ✓SECOND (Ready) devient ✓ALONE (Ready). L'application dans le script start_prim du module est démarrée sur le serveur ALONE avec le message dans le log

  "Script start_prim"
- Sur timeout dans la console web, l'ancien serveur ✓PRIM (Ready) devient !ERROR (connection error)
- Après reboot du serveur arrêté, vérifier que le shutdown de l'OS a réellement été appelé avec un shutdown

  "Action shutdown called by SYSTEM/root"
- Vérifier que le script stop_prim de l'application a été exécuté avec le message

  "Script stop_prim"
- Vérifier que le module a été complètement arrêté avant le shutdown du serveur avec le dernier message

"Local state STOP NotReady"

- Après reboot du serveur arrêté, si le module est automatiquement démarré au boot (`safekit boot status`), message dans le log

"Action start called at boot time"

- Après démarrage du module sur le serveur arrêté, le module devient ✓`SECOND (Ready)` sur ce serveur et ✓`PRIM (Ready)` sur l'autre serveur

4.2.9 Test power-off du serveur ✓`PRIM (Ready)`

En cas de panne de courant, le module n'est pas correctement arrêté comme il le serait lors d'un shutdown du serveur. Le basculement est déclenché par la perte des heartbeats, plutôt que par la détection de l'arrêt du module.

`userconfig.xml` avec 2 heartbeats :

```
<heart>
 <heartbeat name="default" />
 <heartbeat name="private"
            ident="flow" />
</heart>
```

Note : si vous voulez faire un test de double panne électrique simultanée sur les deux serveurs, vérifier que `<rfs async="none">` est positionné dans `userconfig.xml`. Pour plus d'information, voir la section 13.6.3.

- Dans le journal du nœud ✓`SECOND (Ready)`, message

"Resource heartbeat.default set to down by heart"
"Resource heartbeat.flow set to down by heart"
"Remote state UNKNOWN"
"Action alone called by heart : no heartbeat"

- Les messages apparaissent après 30 secondes à la suite du power-off (si aucun timeout configuré dans la section `<heart>` de `userconfig.xml`)
- Le serveur ✓`SECOND (Ready)` devient ✓`ALONE (Ready)` ; l'application dans le script `start_prim` du module est démarrée sur le serveur avec le message dans son log

"Script start_prim"

- Sur timeout de la console web, l'ancien serveur ✓`PRIM (Ready)` devient !`ERROR (connection error)`
- Après reboot du serveur arrêté, si le module est démarré automatiquement au boot (`safekit boot status`), message dans le log

"Action start called at boot time"

- Après redémarrage du module sur le serveur arrêté, le module devient ✓`SECOND (Ready)` sur ce serveur et ✓`PRIM (Ready)` sur l'autre serveur

4.2.10 Test split-brain avec un module miroir

Le split-brain se produit en situation d'isolation réseau entre les deux serveurs SafeKit. Chaque serveur devient primaire `ALONE` et tourne l'application. Au retour du split-brain, un sacrifice doit être réalisé en arrêtant l'application sur un seul des deux serveurs.

Module miroir dans l'état ✓`PRIM (Ready)` et ✓`SECOND (Ready)`

`userconfig.xml` :

```
<heart>
 <heartbeat name="default" />
 <heartbeat name="repli" ident="flow"
/>
</heart>
```

+
sur Windows pour gérer le conflit sur l'adresse IP virtuelle `virtip`

```
<vip>
 <interface_list>
  <interface check="off">
   <real_interface>
    <virtual_addr addr="virtip"
       where="one_side_alias"
       check="on"/>
   </real_interface>
  </interface>
 </interface_list>
</vip>
```

Pour obtenir le split-brain, vérifier qu'il n'y a pas de checkers dans `userconfig.xml` qui peuvent détecter l'isolation : pas de `<interface check="on">` ou de `<ping>` checker

1. Déconnecter en même temps les réseaux `default` et `repli`
2. Reconnecter les réseaux

1. Après isolation réseau des deux serveurs, tous les heartbeats sont perdus. Dans les logs des 2 serveurs,

   "Resource heartbeat.default set to down by heart"
   "Resource heartbeat.flow set to down by heart"
   "Remote state UNKNOWN"
   "Local state ALONE Ready"

   Cas de split-brain : les 2 serveurs sont ✓`ALONE (Ready)` et l'application est démarrée
2. Lorsque les réseaux sont reconnectés, sacrifice d'un des 2 serveurs `ALONE` : l'ancien serveur `SECOND`

   Journal de l'ancien `PRIM` non sacrifié :

   "Remote state ALONE Ready"
   "Split brain recovery: staying alone"

   Journal de l'ancien `SECOND` sacrifié :

   "Remote state ALONE Ready"
   "Split brain recovery: exiting alone"
   "Script stop_prim"

   Le serveur réalise un stopstart : arrêt de l'application avec `stop_prim` puis réintégration des fichiers répliqués à partir de l'autre serveur.

   En Windows, sur reconnexion, il peut se produire un conflit sur l'adresse IP virtuelle, entraînant le `stopstart` du module.
3. Retour à l'état ✓`PRIM (Ready)` et ✓`SECOND (Ready)` sur les 2 serveurs tel qu'ils étaient avant split-brain

Note : la situation de split-brain dans un module miroir avec réplication est malsaine. En effet, le sacrifice de l'ex secondaire provoque la réintégration des données sur ce serveur à partir de la primaire et la perte des données enregistrées sur la secondaire pendant la situation de split-brain.

Pour cette raison, 2 voies de heartbeat sur 2 réseaux physiquement distincts

	sont conseillées. Typiquement, un câble entre les deux serveurs va permettre (1) d'éviter le split-brain avec un réseau supplémentaire de heartbeat et (2) de faire passer le flux de réplication.

4.2.11 Continuer les tests de votre module miroir avec les checkers

Voir les tests décrits en section 4.4.

4.3 Tests d'un module ferme

4.3.1 Test start d'un module ferme sur les serveurs ✗STOP (NotReady)

- Message dans les logs de tous les nœuds (pour lire les journaux, voir la section 7.3)

  "Action start called by admin@<IP>/SYSTEM/root"

- Le module va dans l'état ✓UP (Ready) sur tous les serveurs
- L'application est démarrée dans le script `start_both` du module sur tous les serveurs avec le message dans les logs

  "Script start_both"

4.3.2 Test stop d'un module ferme sur un serveur ✓UP (Ready)

- Message dans le journal du nœud arrêté (pour lire les journaux, voir la section 7.3)

  "Action stop called by admin@<IP>/SYSTEM/root"

- Le nœud qui s'arrête exécute le script `stop_both` pour stopper l'application localement avec le message dans le log

  "Script stop_both"

- Le module sur le serveur arrêté devient ✗STOP (NotReady) avec les messages dans son journal :

  "Local state STOP NotReady"

- L'autre serveur reste ✓UP(Ready) et continue à exécuter l'application
- Redémarrer le serveur ✗STOP (NotReady) avec la commande start

4.3.3 Test restart d'un module ferme sur un serveur ✓UP (Ready)

- Message dans le journal du module où la commande restart est serveurs (pour lire les journaux, voir la section 7.3)

  "Action restart called by admin@<IP>/SYSTEM/root"

- Le module redémarré devient UP (Transient) puis devient ✓UP (Ready)
- Les scripts du module `stop_both`/`start_both` sont exécutés sur le serveur pour redémarrer localement l'application avec les messages dans le log

  "Script stop_both"
  "Script start_both"

4.3.4 Test adresse IP virtuelle d'un module ferme

4.3.4.1 Configuration avec vmac_directed

Module ferme dans l'état ✓`UP (Ready)` sur les 3 serveurs `ip1.1`, `ip1.2`

`userconfig.xml` avec load balancing sur le service `safewebserver` (port TCP 9010) :

```
<farm>
<lan name="default" />
</farm>

<vip>
 <interface_list>
  <interface check="on">
  <virtual_interface type="vmac_directed" >
    <virtual_addr addr="virtip"
where="alias" check="on"/>
  </virtual_interface>
  </interface>
 </interface_list>

<loadbalancing_list>
<group name="FarmProto">
  <rule port="9010" proto="tcp"
filter="on_port"/>
</group>
</loadbalancing_list>
</vip>
```

Sur un poste (ou un serveur) externe dans le même LAN, ping des 2 IP physiques + IP virtuelle + arp -a

- Dans le journal verbeux de tous les serveurs :

  "Virtual IP <virtip of farm> set"

- Sur les 2 serveurs, `ipconfig /all` (Windows) ou `ip addr show` (Linux) retourne `virtip` en alias sur l'interface réseau.
- Sur une station de travail (ou un serveur) du même LAN, les pings répondent et `virtip` est mappée sur l'adresse MAC de l'un des deux serveurs :

```
ping node1_ip_address
ping node2_ip_address
ping virtip

arp -a
node1_ip_address      00-0c-29-0a-5c-fc
node2_ip_address      00-0c-29-26-44-93
virtip      00-0c-29-26-44-93
```

4.3.4.2 Configuration avec vmac_invisible

Module ferme dans l'état ✓`UP (Ready)` sur les 3 serveurs ip1.1, ip1.2

`userconfig.xml` avec load balancing sur le service `safewebserver` (port TCP 9010) :

```
<farm>
<lan name="default" />
</farm>

<vip>
  <interface_list>
  <interface check="on">
  <virtual_interface type="vmac_invisible" >
    <virtual_addr addr="virtip"
where="alias" check="on"/>
  </virtual_interface>
  </interface>
 </interface_list>
```

- Dans le journal verbeux de tous les serveurs :

  "Vitual IP <virtip of farm> set"

- Sur les 2 serveurs, `ipconfig /all` (Windows) ou `ip addr show` (Linux) retourne `virtip` en alias sur l'interface réseau.
- Sur une station de travail (ou un serveur) du même LAN, les pings répondent et `virtip` est mappée sur l'adresse MAC virtuelle :

```
ping node1_ip_address
ping node2_ip_address
ping virtip
```

```
<loadbalancing_list>
<group name="FarmProto">
  <rule port="9010" proto="tcp"
filter="on_port"/>
</group>
</loadbalancing_list>
</vip>
```

Sur un poste (ou un serveur) externe dans le même LAN, ping des 2 IP physiques + IP virtuelle + arp -a

```
arp -a
node1_ip_address      00-0c-29-0a-5c-fc
node2_ip_address      00-0c-29-26-44-93
virtip      5a-fe-c0-a8-38-14
```

- Note : par défaut, l'adresse MAC virtuelle est une adresse Ethernet unicast construite avec `5A:FE` (SAFE) et l'adresse IP virtuelle en hexadécimale

4.3.5 Test load balancing TCP sur une adresse virtuelle

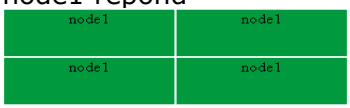
Le module ferme est dans l'état ✓`UP (Ready)` sur les 2 serveurs node1, node2.

Même configuration de load balancing dans `userconfig.xml` que le test précédent.

Sur une station distante :

1. Se connecter à l'URL http://virtip:9010/safekit/mosaic.html, puis cliquer sur Mosaic Test. node1, node2 répondent

2. stop du module sur node2. Rechargement de l'URL. Seul node1 répond



Commande spéciale pour vérifier la bitmap de load balancing sur le port 9010 et sur chaque nœud ✓`UP (Ready)` :

```
safekit -r vip_if_ctrl -l
```

Une entrée de la bitmap de 256 bits doit être à 1 sur un seul serveur

De plus, les 256 bits sont distribués de manière équitable dans les bitmaps de tous les serveurs ✓`UP (Ready)` (si pas de définition de l'attribut `power` dans `userconfig.xml`)

1. ✓`UP (Ready)` sur les 2 serveurs : load balancing des sessions TCP entre node1, node2 en chargeant l'URL
   - Dans les ressources du module, pour node1 et node2 : `FarmProto_0 50%`
   - Journal verbeux de node1 et node2

     "farm membership: **node1 node2** (group FarmProto_0)"
     "farm load: **128/256** (group FarmProto_0)"

     128/256 : 128 bits sur 256 sont gérés par chacun des serveurs
   - `safekit -r vip_if_ctrl -l` sur node1 et node2

     Avec `type="vmac_directed"`

     Bitmap node1:
     01010101:01010101:01010101:01010101:ffffffff:ffffffff:ffffffff:ffffffff
     Bitmap node2:
     ffffffff:ffffffff:ffffffff:ffffffff:02020202:02020202:02020202:02020202

     01 and 02 correspondent au numéro des nœuds qui répondent.

     Avec `type="vmac_invisible"`

     Bitmap node1:
     00000000:00000000:00000000:00000000:ffffffff:ffffffff:ffffffff:ffffffff
     Bitmap node2:
     ffffffff:ffffffff:ffffffff:ffffffff:00000000:00000000:00000000:00000000

     Les bits sont équitablement répartis entre les 2 nœuds.
2. ✗`STOP (NotReady)` sur node2 : les sessions TCP sont servies par node1 lorsqu'on charge l'URL

- Dans les ressources du module, pour node1 : `FarmProto_0 100%`
- Dans le journal verbeux de node1 :

"farm membership: **node1** (group FarmProto_0)"
"farm load: **256/256** (group FarmProto_0)"

256/256 : tous les bits sont gérés par node1

- `safekit -r vip_if_ctrl -l` sur node1:

Bitmap :
ffffffff:ffffffff:ffffffff:ffffffff:ffffffff:ffffffff:ffffffff:ffffffff

4.3.6 Test split-brain avec un module ferme

Le split-brain se produit en situation d'isolation réseau entre les serveurs SafeKit.

Le module ferme est ✓`UP (Ready)` ✓`UP (Ready)` sur node1 et node2.

Même configuration de load balancing dans `userconfig.xml` que les tests précédents. Pour obtenir le split-brain, vérifier qu'il n'y a pas de checker pouvant détecter l'isolation : pas de `<interface check="on">` ou de checker `<ping>`.

Sur la station externe :

1. Se connecter à http://virtip:9010/safekit/mosaic.html, puis cliquer sur Mosaic Test. node1 and node2 répondent

2. Déconnecter le réseau `default` entre node1 et node2. Suivant l'endroit où se trouve la station externe, node 1 ou node 2 répond

ou

1. Avant split-brain, état ✓`UP (Ready)` sur node1 et node2.
   - Dans les ressources pour node1 et node2 : `FarmProto_0 50%`
   - Dans les logs verbeux de node1 et node2 :

"farm membership: **node1 node2** (group FarmProto_0)"
"farm load: **128/256** (group FarmProto_0)"

128/256 : 128 bits sur 256 sont gérés par chacun des serveurs

   - `safekit -r vip_if_ctrl -l` sur node1 et node2 :

Avec `type="vmac_directed"`

Bitmap node1:
01010101:01010101:01010101:01010101:ffffffff:ffffffff:ffffffff:ffffffff
Bitmap node2:
ffffffff:ffffffff:ffffffff:ffffffff:02020202:02020202:02020202:02020202

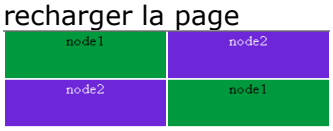
01 and 02 correspondent au numéro des nœuds qui répondent.

Avec `type="vmac_invisible"`

Bitmap node1:
00000000:00000000:00000000:00000000:ffffffff:ffffffff:ffffffff:ffffffff
Bitmap node2:
ffffffff:ffffffff:ffffffff:ffffffff:00000000:00000000:00000000:00000000

Les bits sont équitablement répartis entre les 2 serveurs

3. Reconnecter le réseau et se recharger la page



Même commande spéciale que le test précédent pour vérifier la bitmap de load balancing sur le port 9010 sur chaque nœud ✓`UP (Ready)`

2. Après isolation réseau, split-brain :
   - Dans les ressources, pour node1 et node2 : `FarmProto_0 100%`
   - Dans le journal verbeux de node1 :

     "farm membership: **node1** (group FarmProto_0)"
     "farm load: **256/256** (group FarmProto_0)"

     256/256 : tous les bits sont gérés par node1
   - Dans le journal verbeux de node2 :

     "farm membership: **node2** (group FarmProto_0)"
     "farm load: **256/256** (group FarmProto_0)"

     256/256 : tous les bits sont gérés par node2
   - `safekit -r vip_if_ctrl -l` sur node1 et node2:

     Bitmap:
     ffffffff:ffffffff:ffffffff:ffffffff:ffffffff:ffffffff:ffffffff:ffffffff:ffffffff
3. Après split-brain lorsque le réseau est reconnecté, les mêmes messages peuvent être trouvés dans le journal et les mêmes bitmaps que ceux avant split-brain

Le comportement par défaut d'une ferme en situation de split-brain est correct. La recommandation est de mettre dans `userconfig.xml` un réseau de surveillance `<lan> </lan>`, là où se trouve l'adresse IP virtuelle.

Les messages dans le journal et le résultat de `vip_if_ctrl` sont légèrement différents selon le type `vmac_directed` ou `vmac_invisible`.

4.3.7 Test de la compatibilité du réseau avec l'adresse MAC invisible (vmac_invisible)

Prérequis réseau

Une adresse MAC Ethernet unicast 5a-fe-xx-xx-xx-xx est associée à l'adresse virtuelle `virtip` d'un module ferme. Elle n'est jamais présentée par les serveurs SafeKit en tant qu'adresse Ethernet source (MAC invisible). Les switchs ne peuvent donc pas localiser cette adresse. Lorsqu'ils font suivre un paquet à destination de l'adresse MAC 5a-fe-xx-xx-xx-xx, ils doivent broadcaster le paquet sur tous les

1. Tous les serveurs sont ✓`UP (Ready)`
2. Le monitoring réseau est lancé dans chaque serveur en filtrant sur `virtip`
3. Un terminal externe envoie un seul ping vers l'adresse IP virtuelle avec `ping -n (ou -c) 1 virtip`
   - 1 paquet envoyé et reçu par l'ensemble des serveurs

     "ICMP: Echo: From extip To virtip"
   - il doit y avoir autant de paquets

     "ICMP: Echo Reply: To extip From virtip"

ports du LAN ou VLAN où se situe l'adresse IP virtuelle (flooding). Et tous les serveurs de la ferme reçoivent donc les paquets à destination de l'adresse MAC virtuelle 5a-fe-xx-xx-xx-xx.

Noter que ce prérequis n'existe pas pour un module miroir (voir section 4.2.6).

Prérequis serveur

Les paquets remontent dans les cartes Ethernet mises en mode promiscuous par SafeKit. Et les paquets sont filtrés par le module kernel <vip> suivant la bitmap de load balancing. Pour réaliser le test, il faut un outil de monitoring du réseau.

Ex. monitoring réseau sur Linux :

```
tcpdump host virtip
```

qu'il y a de serveurs ✓UP (Ready)

Si ce n'est pas le cas, vérifier si des options restreignent le "port flooding" dans les switchs et empêche le broadcast de "ICMP: Echo" vers tous les serveurs.

Attention : la restriction "port flooding" dans les switchs peut avoir lieu après un certain nombre de flooding (temps, nombre de KB floodés) : le test ping est à répéter sur plusieurs heures en créant du flooding sur l'adresse IP virtuelle

Note : pour éviter les outils de monitoring réseau, une console externe Linux peut être utilisée. Le ping Linux permet de vérifier les paquets dupliqués revenant des 3 serveurs ✓UP (Ready) :

```
ping virtip
64 bytes from virtip icmp_seq=1
64 bytes from virtip icmp_seq=1 (DUP!)
64 bytes from virtip icmp_seq=1 (DUP!)
64 bytes from virtip icmp_seq=2
64 bytes from virtip icmp_seq=2 (DUP!)
64 bytes from virtip icmp_seq=2 (DUP!)
...
```

Ce test peut être réalisé pendant plusieurs minutes en stockant l'output du ping dans un fichier et en vérifiant ensuite qu'il y a eu des (DUP!) tout le temps : `date > /tmp/ping.txt ; ping virtip >> /tmp/ping.txt`

4.3.8 Test shutdown d' un serveur ✓UP (Ready)

- Sur Windows, vérifier que la procédure spéciale d'arrêt des modules au shutdown a été réalisée : voir section 10.4
- Effectuer un shutdown d'un serveur ✓UP (Ready)
- Les autres serveurs restent ✓UP (Ready) et continuent à exécuter l'application
- Sur timeout dans la console web, l'ancien serveur ✓UP (Ready) devient !ERROR (connection error)
- Après reboot du serveur arrêté, vérifier que le shutdown de l'OS a réellement appelé un shutdown du module avec le message

```
"Action shutdown called by SYSTEM"
```

- Vérifier que le script `stop_both` de l'application a été exécuté avec le message

```
"Script stop_both"
```

- Vérifier que le module a été complètement arrêté avant le shutdown du serveur avec le dernier message

```
"Local state STOP NotReady"
```

- Après reboot du serveur arrêté, si le module est automatiquement démarré au boot (`safekit boot status`), message dans le log

  "Action start called at boot time"

- Après démarrage du module sur le serveur arrêté, le module devient ✓`UP (Ready)` sur ce serveur et il exécute le script `start_both` qui redémarre l'application sur le serveur avec le message dans le log

  "Script start_both"

4.3.9 Test power-off d'un serveur ✓UP (Ready)

En cas de panne de courant, le module n'est pas correctement arrêté comme il le serait lors d'un shutdown du serveur. Le basculement est déclenché par la perte des heartbeats, plutôt que par la détection de l'arrêt du module

- Les autres serveurs restent ✓`UP (Ready)` et continuent à exécuter l'applicatif
- Sur timeout dans la console web, l'ancien serveur ✓`UP (Ready)` devient !`ERROR (connection error)`
- Après reboot du serveur arrêté, si le module est démarré automatiquement au boot (`safekit boot status`), message dans le log

  "Action start called at boot time"

- Après redémarrage du module sur le serveur rebooté, le module devient ✓`UP (Ready)` et il exécute le script `start_both` qui relance l'applicatif sur ce serveur avec le message dans le log

  "Script start_both"

4.3.10 Continuer les tests du module ferme avec les checkers

Voir les tests décrits en la section 4.4.

4.4 Tests des checkers communs à un miroir et une ferme

4.4.1 Test <errd> checker avec action restart ou stopstart

Pour la description de la surveillance de processus/service, voir la section 13.9.

Dans `userconfig.xml` :

```
<errd>
 <proc name="appli.exe" atleast="1"
action="restart" class="prim"/>
</errd>
```

Le checker surveille le processus nommé `appli.exe`.

- `name="appli.exe" atleast="1"`

  Au moins un processus `appli.exe` doit s'exécuter

- `class`
  - `class="prim"` pour un module miroir

1. Tuer le processus `appli.exe` sur le serveur ✓`(Ready)` ; c'est-à-dire dans l'état `PRIM` ou `ALONE` pour un module miroir et l'état `UP` pour un module ferme.
   - Messages dans le journal :

     "Process appli.exe not running"
     "Action restart|stopstart called by errd"

   - Le module passe dans un état 🔄 `(Transient)`, respectivement `PRIM`, `ALONE` ou `UP`
   - Dans le cas `restart`, le module redevient ✓`(Ready)`,

Checker démarré/arrêté, sur le serveur dans l'état ✓PRIM ou ALONE(Ready), après/avant l'application (start_prim /stop_prim)

- o class="both" pour un module ferme

  Checker démarré/arrêté, sur tous les serveurs dans l'état ✓UP (Ready), après/avant l'application (start_both /stop_both)

- action

  Si le processus est absent, le checker met la ressource proc.appli.exe à down. Puis il exécute un restart ou stopstart.

  - o action="restart"

    L'application est redémarrée localement (stop_xx ; start_xx)

  - o action="stopstart"

    Le module est arrêté, ainsi que l'application, puis est démarré automatiquement

respectivement dans l'état PRIM, ALONE ou UP

- o Dans le cas stopstart, le module redevient ✓(Ready), respectivement dans l'état SECOND, ALONE ou UP

  Message dans le journal :

  "Action start called automatically"

  Note : un stopstart sur ✓PRIM (Ready) provoque un basculement

2. Reproduire le test sur le même serveur s'il tourne toujours l'application (i.e. ✓(Ready) ALONE, PRIM ou UP) :

   Par défaut, à la 4ème détection d'erreur en 24h (voir maxloop et loop_interval décrits dans la section 13.2.3), le module va dans l'état ✗ STOP (NotReady).

   Message dans le journal avant l'arrêt :

   "Action stop called by maxloop"

4.4.2 Test <tcp> checker avec action restart ou stopstart

Pour la description du TCP checker, voir la section 13.11.

Dans userconfig.xml :

```
<check>
  <tcp ident="id" when="prim"
action="restart" >
    <to addr="addr" port="port"/>
  </tcp>
</chek>
```

Le checker vérifie que l'application répond aux demandes de connexion.

- addr="addr", port="port"

  Connexions TCP testées addr:port

- when

  - o when="prim" pour un module miroir

    Checker démarré/arrêté, sur le serveur dans l'état ✓PRIM ou ALONE(Ready), après/avant

1. Arrêter l'application qui écoute sur addr:port sur le serveur dans un état ✓(Ready) ; c'est-à-dire dans l'état PRIM ou ALONE pour un module miroir et l'état UP pour un module ferme.

   - o Messages dans le journal :

     "Resource tcp.id set to down by tcpcheck"
     "Action restart|stopstart from failover rule t_id "

   - o Le module passe dans un état ↻ (Transient

   - o Dans le cas restart, le module redevient ✓(Ready), respectivement dans l'état PRIM, ALONE ou UP

   - o dans le cas stopstart, le module redevient ✓(Ready),

l'application (`start_prim` /`stop_prim`)

- `when="both"` pour un module ferme

  Checker démarré/arrêté, sur tous les serveurs dans l'état ✓`UP (Ready)`, après/avant l'application (`start_both` /`stop_both`)

- `action`

  Si la connexion échoue, le checker met la ressource `tcp.id` à `down`. La règle de failover associée, nommée `t_id`, exécute un `restart` ou `stopstart`.

  - `action="restart"`

    L'application est redémarrée localement (`stop_xx ; start_xx`)

  - `action="stopstart"`

    Le module est arrêté, ainsi que l'application, puis est démarré automatiquement

respectivement dans l'état `SECOND`, `ALONE` ou `UP`

Message dans le journal :

"Action start called automatically"

Note : un `stopstart` sur ✓`PRIM (Ready)` provoque un basculement

2. Reproduire le test sur le même serveur s'il tourne toujours l'application (i.e. ✓`(Ready) ALONE` ou `UP`).

   Par défaut, à la 4ème détection d'erreur en 24h (voir `maxloop` et `loop_interval` décrits dans la section 13.2.3), le module va dans l'état ✗ `STOP (NotReady)`.

   Message dans le journal avant l'arrêt :

   "Action stop called by maxloop"

4.4.3 Test <tcp> checker avec action wait

Pour la description du TCP checker, voir la section 13.11.

Dans `userconfig.xml` :

```
<check>
  <tcp ident="id" when="pre"
action="wait" >
    <to addr="addr" port="port"/>
  </tcp>
</check>
```

Le checker vérifie qu'une application, externe au module, répond aux demandes de connexion.

- `addr="addr"`, `port="port"`

  Connexions TCP testées `addr:port`

- `when="pre"`

  Checker démarré avant, arrêté après, l'application intégrée dans le module (dans `start_xx` /`stop_xx`)

- `action="wait"`

  Si la connexion échoue, le checker met la ressource `tcp.id` à `down`. La règle

1. Arrêter l'application externe qui écoute sur `addr:port` quand le serveur est ✓`(Ready)` ; c'est-à-dire dans l'état `PRIM`, `ALONE` ou `SECOND` pour un module miroir et l'état `UP` pour un module ferme.

   - Messages dans le journal :

     "Resource tcp.id set to down by tcpcheck"
     "Action wait from failover rule t_id"

   - Le module devient ○`WAIT (NotReady)` sur tous les nœuds

     Note : un `wait` sur ✓`PRIM (Ready)` provoque un basculement

2. Redémarrer l'application externe qui écoute sur `addr:port`

   - Messages dans le journal verbeux

     "Resource tcp.id set to up by tcpcheck"
     "Action wakeup from failover rule Implicit_wakeup"

de failover associée, nommée `t_id`, exécute un `wait`.

Cela arrête le module, et son application, puis l'amène dans l'état `WAIT` en attente de `tcp.id` repositionné à `up` par le checker.

- le module redevient ✓ `(Ready)`

3. Répéter le test.

   Par défaut, à la 4ème détection d'erreur en 24h (voir `maxloop` et `loop_interval` décrits dans la section 13.2.3), le module va dans l'état ✗ `STOP (NotReady)`.

   Message dans le journal avant l'arrêt :

   "Action stop called by maxloop"

Note : ce test permet de tester la connectivité d'un serveur à un service externe. Mais si le service externe est en panne ou s'il est inatteignable sur tous les serveurs, tous les serveurs vont en ○ `WAIT (NotReady)` et l'application est indisponible.

4.4.4 Test <interface check="on"> avec action wait

Pour la description du checker d'interface, voir la section 13.13. Pour sa configuration automatique avec `<interface check="on">`, voir la section 13.5.5.

Dans `userconfig.xml` :

```
<vip>
 <interface_list>
 <interface check="on">
 <real_interface>
   <virtual_addr addr="172.17.0.20"
              where="one_side_alias"
             check="on"/>
</real_interface>
  </interface>
 </interface_list>
</vip>
```

Le checker vérifie que le câble Ethernet est connecté sur l'interface du réseau où est définie l'adresse IP virtuelle.

- Si une erreur est détectée sur l'interface, le checker met la ressource associée `intf.172.17.0.0` à `down`. Le préfixe est `intf` et le suffixe est le réseau correspondant à l'IP virtuelle.
- La règle de failover par défaut, nommée `interface_failure`, exécute un `wait`.

  Cela arrête le module, et son application, puis l'amène dans l'état

1. Retirer le câble Ethernet de la carte du réseau (sur laquelle est configurée l'IP virtuelle) sur le serveur dans un état ✓ `(Ready)` ; c'est-à-dire dans l'état `PRIM` ou `ALONE` pour un module miroir et l'état `UP` pour un module ferme :
   - Messages dans le journal :

     "Resource intf.172.17.0.0 set to down by intfcheck"
     "Action wait from failover rule interface_failure"

   - Le module devient ○ `WAIT (NotReady)`

     Note : un `wait` sur ✓ `PRIM (Ready)` provoque un basculement

2. Remettre le câble
   - Messages dans le journal verbeux

     "Resource intf.172.17.0.0 set to up by intfcheck"
     "Action wakeup from failover rule Implicit_wakeup"

   - Le module redevient ✓ `(Ready)`, respectivement dans l'état `SECOND`, `ALONE` ou `UP`

WAIT en attente de intf.172.17.0.0 repositionné à up par le checker.

Note : ne pas utiliser check="on" sur des interfaces de bonding ou teaming car ces interfaces apportent leurs propres mécanismes de reprise d'interface à interface

Note : un wait sur ✓PRIM (Ready) provoque un basculement

3. Reproduire le test sur le même serveur

   Par défaut, à la 4ème détection d'erreur en 24h (voir maxloop et loop_interval décrits dans la section 13.2.3), le module va dans l'état ✗ STOP (NotReady).

   Message dans le journal avant l'arrêt :

   "Action stop called by maxloop"

Note : Désactiver l'interface au lieu de débrancher le câble réseau conduit à l'état ✗ (NotReady) STOP si ce réseau est utilisé comme heartbeat. Dans ce cas, le module ne peut démarrer (ni redémarrer) car l'adresse IP locale, pour le heartbeat, n'est pas définie.

4.4.5 Test <ping> checker avec action wait

Pour la description du ping checker, voir la section 13.12.

Dans userconfig.xml:

```
<check>
  <ping ident="id" when="pre"
action="wait">
    <to addr="extip"/>
  </ping>
</check>
```

Le checker vérifie qu'un composant externe (ex. : un routeur) avec l'adresse extip répond au ping.

- when="pre"

  Checker démarré avant, arrêté après, l'application intégrée dans le module (dans start_xx /stop_xx)

- action="wait"

  Si le ping échoue, le checker met la ressource ping.id à down. La règle de failover associée, nommée p_id, exécute un wait.

  Cela arrête le module, et son application, puis l'amène dans l'état WAIT en attente de ping.id repositionné à up par le checker.

1. Rompre la liaison réseau entre le composant externe testé et un serveur dans un état ✓(Ready) ; c'est-à-dire dans l'état PRIM, ALONE ou SECOND pour un module miroir et l'état UP pour un module ferme.
   - Messages dans le journal :

     "Resource ping.id set to down by pingcheck"
     "Action wait from failover rule p_id"

   - Le module devient ○WAIT (NotReady) sur tous les nœuds

     Note : un wait sur ✓PRIM (Ready) provoque un basculement

2. Rétablir la liaison réseau
   - Messages dans le journal verbeux

     "Resource ping.id set to up by pingcheck"
     "Action wakeup from failover rule Implicit_wakeup"

   - le module redevient ✓(Ready)

3. Répéter le test.

   Par défaut, à la 4ème détection d'erreur en 24h (voir maxloop et

	loop_interval décrits dans la section 13.2.3), le module va dans l'état ✗ STOP (NotReady). Message dans le journal avant l'arrêt : "Action stop called by maxloop" Note : ce test permet de tester la connectivité d'un serveur au réseau. Mais si le composant externe est en panne et si le ping échoue sur tous les serveurs, tous les serveurs vont en ○ WAIT (NotReady) et l'application est indisponible.

4.4.6 Test <module> checker avec action wait

Pour la description du module checker, voir la section 13.16.

In userconfig.xml du module AM :

```
<check>
  <module name="othermodule">
    <to addr="ip" addr="9010"/>
  </module>
</check>
```

Le checker du module AM, vérifie que le module othermodule, accédé depuis son adresse IP virtuelle ip, est ✓ (Ready).

- Si le module othermodule n'est pas démarré, le checker met la ressource associée module.othermodule_ip à down. Le préfixe est module et le suffixe est le nom du module externe suivi de son adresse.
- La règle de failover par défaut, nommée module_failure, exécute un wait.

  Cela arrête le module, et son application, puis l'amène dans l'état WAIT en attente de module.othermodule_ip repositionné à up par le checker.

Note : si le module AM est un module miroir qui utilise la réplication de fichiers et en raison de la règle notuptodate_server, vous pouvez rencontrer un comportement incorrect avec le module AM bloqué dans un état WAIT si l'action stopstart arrive pendant la transition SECOND vers ALONE

1. Arrêter le module othermodule. Et démarrer le module AM sur tous ses serveurs.
   - Messages dans le journal

     "Resource module.othermodule_ip set to down by modulecheck"
     "Action wait from failover rule module_failure"
   - Le module devient ○ WAIT (NotReady) sur tous les nœuds

     Note : un wait sur ✓ PRIM (Ready) provoque un basculement
2. Démarrer le module othermodule.
   - Messages dans le journal verbeux

     "Resource module.othermodule_ip set to up by modulecheck"
     "Action wakeup from failover rule Implicit_wakeup"
   - Le module redevient ✓ (Ready) sur tous les nœuds
3. Exécuter un restart du module othermodule.
   - Message dans le journal du module AM :

     "stopstart called by modulecheck"
   - Le module AM s'arrête puis démarre automatiquement

4. Répéter le test.

   Par défaut, à la 4ème détection d'erreur en 24h (voir `maxloop` et `loop_interval` décrits dans la section 13.2.3), le module va dans l'état ✗ `STOP (NotReady)`.

   Message dans le journal avant l'arrêt :

   "Action stop called by maxloop"

4.4.7 Test <custom> checker avec action wait

Pour la description du custom checker, voir la section 13.15.

Dans `userconfig.xml` :

```
<check>
  <custom ident="id" when="pre"
exec="customscript" action="wait" />
  </custom>
</check>
```

Le custom checker est une boucle infinie qui effectue un test et affecte la ressource associée à `up` ou `down`, en fonction du résultat du test.

- `when="pre"`

  Checker démarré avant, arrêté après, l'application intégrée dans le module (dans `start_xx` /`stop_xx`)

- `exec="customscript"`

  Script localisé sous *AM*`/bin/`, qui affecte la ressource `custom.id` :

  - Sur échec

    `SAFE/safekit set -r custom.id -v down -i customscript`

  - Sur succès

    `SAFE/safekit set -r custom.id -v up -i customscript`

- `action="wait"`

  Quand la ressource `custom.id` passe à `down`, la règle de failover associée, nommée `c_id`, exécute un `wait`.

  Cela arrête le module, et son application, puis l'amène dans l'état

1. Provoquer l'échec du test du custom checker quand le serveur est dans un état ✓ `(Ready)` ; c'est-à-dire dans l'état `PRIM`, `ALONE` ou `SECOND` pour un module miroir et l'état `UP` pour un module ferme.

   - Messages dans le journal verbeux:

     "Resource custom.id set to down by customscript"
     "Action wait from failover rule c_id"

   - Le module devient ○ `WAIT (NotReady)` sur tous les nœuds

     Note : un `wait` sur ✓ `PRIM (Ready)` provoque un basculement

2. Réparer l'erreur testée par le custom checker.

   - Messages dans le journal verbeux

     "Resource custom.id set to up by customscript "
     "Action wakeup from failover rule Implicit_wakeup"

   - Le module redevient ✓ `(Ready)`

3. Répéter le test.

   Par défaut, à la 4ème détection d'erreur en 24h (voir `maxloop` et `loop_interval` décrits dans la section 13.2.3), le module va dans l'état ✗ `STOP (NotReady)`.

   Message dans le journal avant l'arrêt :

   "Action stop called by maxloop"

WAIT en attente de custom.id repositionné à up par le checker.	

L'action associée au custom checker peut-être définie via une règle de failover explicite plutôt que l'attribut action, qui vaut dans ce cas noaction. L'exemple suivant est équivalent au précédent, excepté pour le nom de la règle de failover qui est customid_failure :

```
<check>
  <custom ident="id" when="pre" exec="customscript" action="noaction" />
  </custom>
</check>
<failover>
  <![CDATA[
   customid_failure: if (custom.id == down) then wait();
  ]]>
</failover>
```

Cette syntaxe est celle supportée avant SafeKit 8.

4.4.8 Test <custom> checker avec action restart ou stopstart

Pour la description du custom checker, voir la section 13.15.

4.4.8.1 Action via une règle de failover

Dans userconfig.xml :

```
<check>
  <custom ident="id" when="prim"
exec="customscript" action="restart"
/>
  </custom>
</check>
```

Le custom checker est une boucle infinie qui effectue un test et affecte la ressource associée à up ou down, en fonction du résultat du test.

- when
  - when="prim" pour un module miroir

    Checker démarré/arrêté, sur le serveur dans l'état ✓ PRIM ou ALONE (Ready), après/avant l'application (start_prim /stop_prim)
  - when="both" pour un module ferme

    Checker démarré/arrêté, sur tous les serveurs dans l'état ✓ UP (Ready), après/avant

1. Provoquer l'échec du test du custom checker le custom checker quand le serveur est dans un état ✓ (Ready) ; c'est-à-dire dans l'état PRIM ou ALONE pour un module miroir et l'état UP pour un module ferme :
   - Messages dans le journal verbeux :

     "Resource custom.id set to down by customscript"

     et

     "Action restart from failover rule c_id "

     ou

     "Action stopstart from failover rule c_id "
   - Le module passe dans un état ↻ (Transient).
   - Dans le cas restart, le module redevient ✓ (Ready), respectivement dans l'état PRIM, ALONE ou UP
   - Dans le cas stopstart, le module redevient ✓ (Ready), respectivement dans l'état SECOND, ALONE ou UP.

     Message dans le journal :

     "Action start called automatically"

l'application (`start_both /stop_both`)

- `exec="customscript"`

  Script localisé sous *AM*`/bin/`, qui affecte la ressource `custom.id` :

  - Sur échec

    ```
    SAFE/safekit set -r custom.id -v down -i customscript
    ```

  - Sur succès

    ```
    SAFE/safekit set -r custom.id -v up -i customscript
    ```

- `action`

  Quand la ressource `custom.id` passe à `down`, la règle de failover, nommée `c_id`, exécute un `restart` ou `stopstart`.

  - `action="restart"`

    L'application est redémarrée localement (`stop_xx ; start_xx`)

  - `action="stopstart"`

    Le module est arrêté, ainsi que l'application et le checker, puis est démarré automatiquement.

Note : un `stopstart` sur ✓`PRIM (Ready)` provoque un basculement

2. Reproduire le test sur le même serveur s'il tourne toujours l'application (i.e. ✓`(Ready) ALONE` ou `UP`).

   Par défaut, à la 4ème détection d'erreur en 24h (voir `maxloop` et `loop_interval` décrits dans la section 13.2.3), le module va dans l'état ✗`STOP (NotReady)`.

   Message dans le journal avant l'arrêt :

   ```
   "Action stop called by maxloop"
   ```

L'action associée au custom checker peut-être définie via une règle de failover explicite plutôt que l'attribut `action`, qui vaut dans ce cas `noaction`. L'exemple suivant est équivalent au précédent, excepté pour le nom de la règle de failover qui est `customid_failure` :

```
<check>
  <custom ident="id" when="pre" exec="customscript" action="noaction" />
  </custom>
</check>
<failover>
  <![CDATA[
   customid_failure: if (custom.id == down) then restart();
  ]]>
</failover>
```

Cette syntaxe est celle supportée avant SafeKit 8.

4.4.8.2 Action directe dans le script

Dans `userconfig.xml` :

```
<check>
```

1. Provoquer l'échec du test du custom checker le custom checker quand le serveur est dans un état ✓`(Ready)` ;

```
  <custom ident="id" when="prim"
exec="customscript" action="noacion"
/>
  </custom>
</check>
```

Le custom checker est une boucle infinie qui effectue un test et effectue un `restart` ou `stopstart`, en fonction du résultat du test.

- `when`
  - `when="prim"` pour un module miroir

    Checker démarré/arrêté, sur le serveur dans l'état ✓`PRIM` ou `ALONE(Ready)`, après/avant l'application (`start_prim` /`stop_prim`)
  - `when="both"` pour un module ferme

    Checker démarré/arrêté, sur tous les serveurs dans l'état ✓`UP (Ready)`, après/avant l'application (`start_both` /`stop_both`)
- `action="noaction"`

  Pas de règle de failover générée.
- `exec="customscript"`

  Script localisé sous *AM*`/bin/`, qui affecte la ressource `custom.id` :
  - Sur échec

    `SAFE/safekit restart -i customscript`

    L'application est redémarrée localement (`stop_xx ; start_xx`)

  Ou
  - Sur échec

    `SAFE/safekit stopstart -i customscript`

    Le module est arrêté, ainsi que l'application et le checker, puis est démarré automatiquement.

c'est-à-dire dans l'état `PRIM` ou `ALONE` pour un module miroir et l'état `UP` pour un module ferme :

- Messages dans le journal verbeux :

  "Action restart called by customscript"

  ou

  "Action stopstart called by customscript"
- Le module passe dans un état 🔄 `(Transient)`.
- Dans le cas `restart`, le module redevient ✓`(Ready)`, respectivement dans l'état `PRIM`, `ALONE` ou `UP`
- Dans le cas `stopstart`, le module redevient ✓`(Ready)`, respectivement dans l'état `SECOND`, `ALONE` ou `UP`.

  Message dans le journal :

  "Action start called automatically"

  Note : un `stopstart` sur ✓`PRIM (Ready)` provoque un basculement

2. Reproduire le test sur le même serveur s'il tourne toujours l'application (i.e. ✓`(Ready) ALONE` ou `UP`)

   Par défaut, à la 4ème détection d'erreur en 24h (voir `maxloop` et `loop_interval` décrits dans la section 13.2.3), le module va dans l'état ✗`STOP (NotReady)`.

   Message dans le journal avant l'arrêt :

   "Action stop called by maxloop"

Note : sur une action directe dans le custom checker, le compteur `maxloop` est incrémenté si `-i identité` est passé à la commande `restart` ou `stopstart`. Sans identité, SafeKit considère qu'il s'agit d'une opération d'administration. Le compteur est remis à 0 et il n'y a pas de stop au bout de 4 redémarrages.

5. Administration d'un module miroir

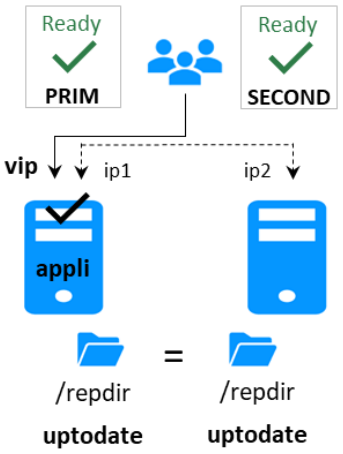
Pour tester un module miroir, voir la section 4.2.

Pour analyser un problème, voir la section 7.

5.1 Mode de fonctionnement d'un module miroir

1. Fonctionnement normal

État stable : primaire avec secondaire

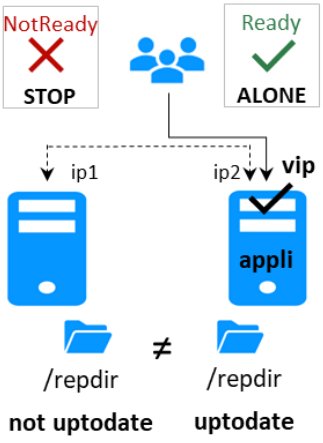


Sur la primaire :

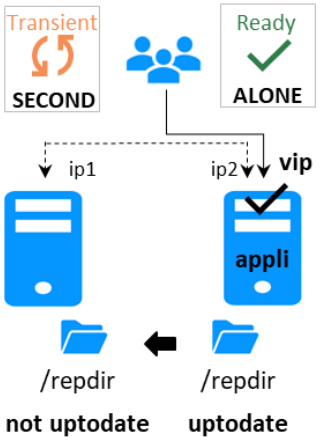
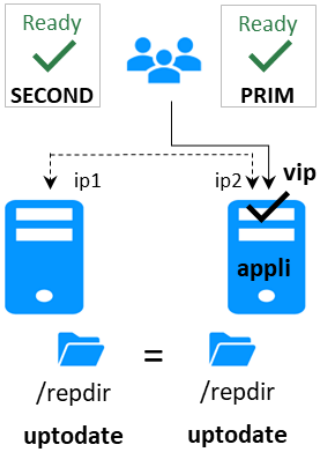
- Réplication de fichiers temps réel
- IP virtuelle définie
- application démarrée

2. Reprise automatique

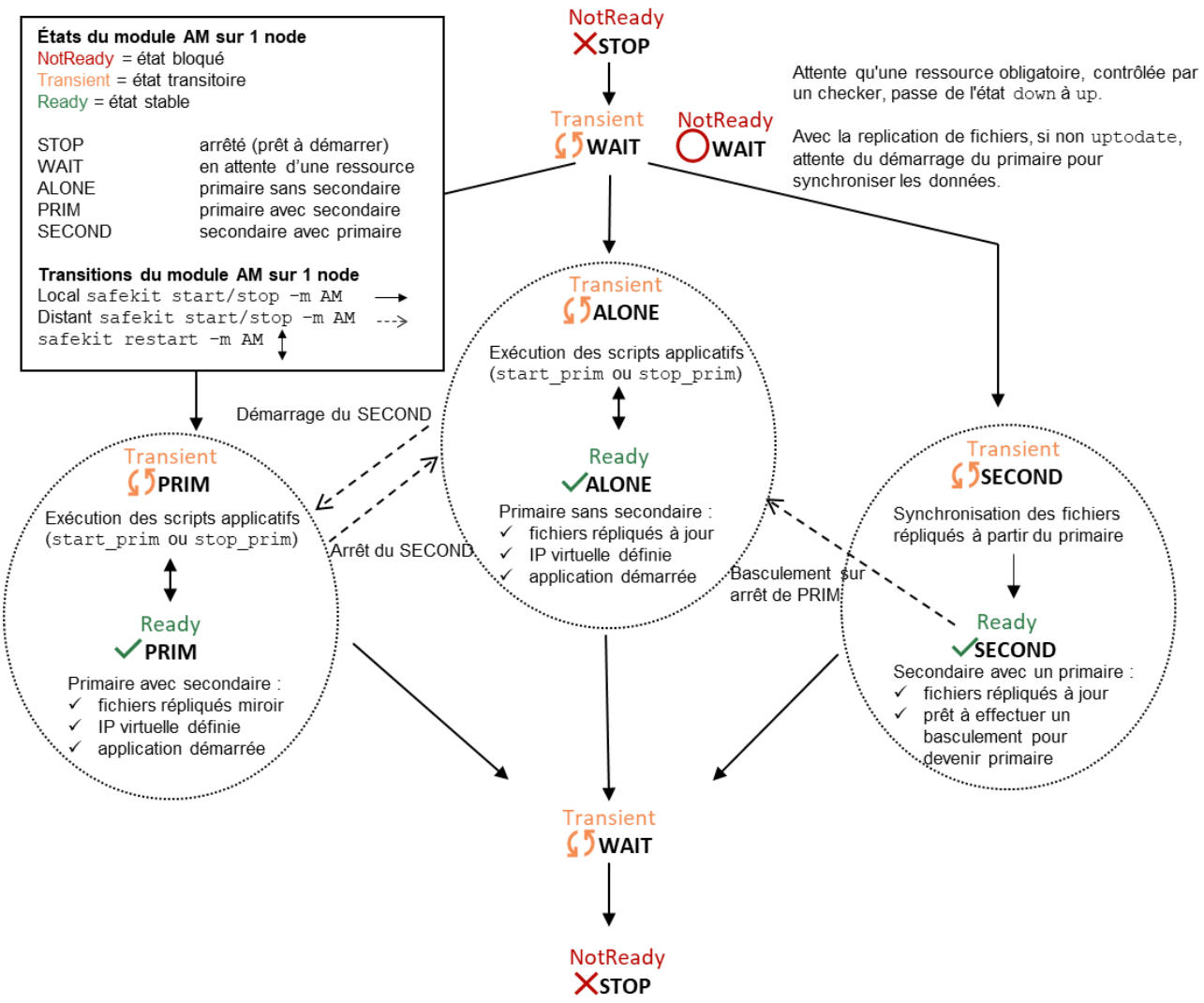
État stable : primaire sans secondaire



Sur arrêt du primaire, reprise automatique de l'IP virtuelle et de l'application.

La secondaire est prête à effectuer une reprise automatique pour devenir primaire.	
3. Réintégration après panne État transitoire : secondaire en cours de réintégration.  Synchronisation automatique des fichiers sans arrêt de l'application et mise à jour uniquement des fichiers modifiés sur le nœud primaire pendant que l'autre nœud était arrêté.	**4. Retour à la normale** État stable : primaire avec secondaire. 

5.2 Automate d'état d'un module miroir (STOP, WAIT, ALONE, PRIM, SECOND - NotReady, Transient, Ready)



5.3 Premier démarrage d'un module miroir (commande `prim`)

Au premier démarrage d'un module miroir, si les deux serveurs sont démarrés avec la commande `start`, ils se bloquent tous les deux dans l'état ○ `WAIT (NotReady)` avec le message dans le journal :

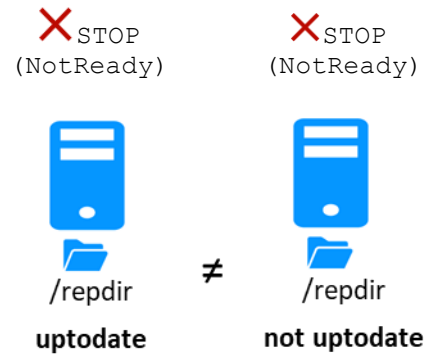
"Data may be not uptodate for replicated directories (wait for the start of the remote server)"

Au premier démarrage, il faut utiliser la commande `prim` pour démarrer en primaire le serveur avec les répertoires à jour, afin de les synchroniser sur l'autre serveur. Ce dernier est démarré avec la commande `second`.

Aux démarrages suivants, utiliser la commande `start` pour démarrer les serveurs.

1. État initial

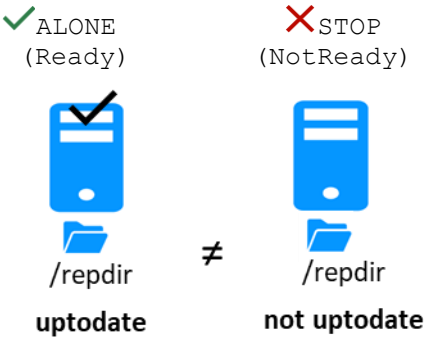
- le module miroir vient juste d'être configuré avec un nouveau répertoire à répliquer entre node1 et node2
- node1 a le répertoire à jour
- node2 a le répertoire vide



2. Commande `prim` sur serveur 1

- utiliser la commande spéciale `safekit prim -m AM` (où *AM* est le nom du module) pour forcer node1 à démarrer en primaire
- pour les démarrages suivants, utiliser toujours de préférence `safekit start -m AM` (où *AM* est le nom du module) : voir section 5.5
- message dans le log:

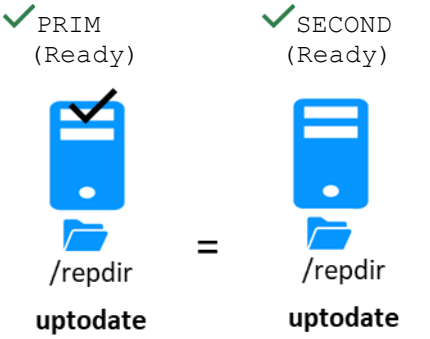
"Action prim called by admin@<IP>/SYSTEM/root"



3. Commande `second` sur node2

- démarrer l'autre serveur en tant que secondaire
- le secondaire réintègre le répertoire répliqué à partir du primaire
- message dans le journal :

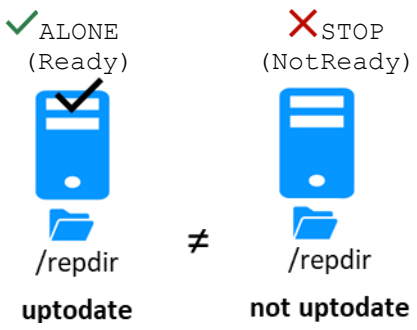
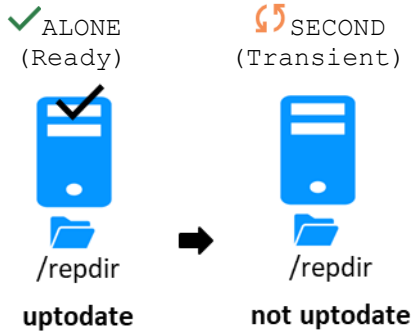
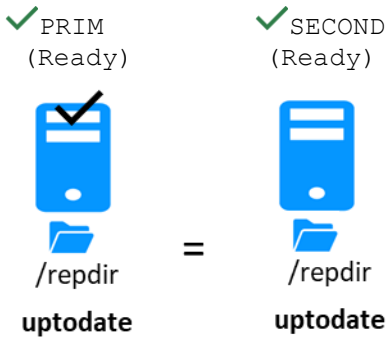
"Action second called by admin@<IP>/SYSTEM/root"



5.4 Différents cas de réintégration (utilisation des bitmaps)

Pour optimiser la réintégration de fichiers, il y a plusieurs cas de figure :

1. Le module doit avoir effectué une réintégration (au premier démarrage du module la réintégration est complète) avant d'activer la gestion des bitmaps de modification
2. Si le module a été proprement arrêté sur le serveur, alors au redémarrage du secondaire, seules les zones modifiées à l'intérieur des fichiers sont réintégrées suivant les bitmaps de modification
3. Si le serveur a crashé (power off), ou a été incorrectement arrêté (exception du processus de réplication `nfsbox`), ou si les fichiers ont été modifiés pendant l'arrêt de SafeKit, les bitmaps de modification ne sont pas sûres et elles ne sont donc pas utilisées. Tous les fichiers qui ont été modifiés pendant et avant l'arrêt suivant une période de grâce (typiquement une heure) sont réintégrés
4. Un appel à la commande spéciale `second fullsync` provoque une réintégration complète de tous les répertoires répliqués sur la secondaire quand elle est redémarrée.

1. Le serveur2 secondaire a été arrêté • les données sont désynchronisées	
2. Commande `start` sur node2 • les données sont réintégrées avec l'optimisation des bitmaps (voir ci-dessus)	
3. Fin de la réintégration • les données sont les mêmes sur les 2 serveurs • seules les modifications à l'intérieur des fichiers sont répliquées en temps réel et de manière synchrone	

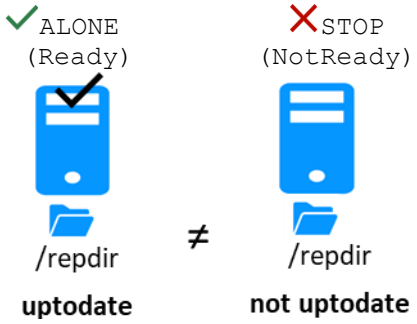
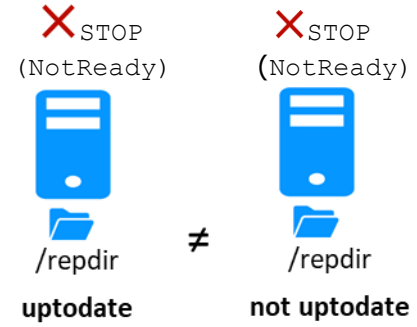
Le système de réplication maintient en plus sur chaque nœud la dernière date à laquelle les données étaient synchronisées. Cette date de synchronisation, nommée `synctimestamp`, est affectée à l'issue de la réintégration et évolue dans l'état ✓`PRIM (Ready)` et ✓`SECOND (Ready)`. Quand le module est arrêté sur le nœud secondaire puis redémarré, la date de synchronisation est un des critères de réintégration : tous les fichiers modifiés autour de cette date sont potentiellement non à jour sur la secondaire et doivent être réintégrés. Depuis SafeKit 7.4.0.50, la date de synchronisation est aussi exploitée pour implémenter une sécurité supplémentaire. Lorsque l'écart entre la date de synchronisation stockée sur la primaire et celle stockée sur la secondaire est supérieur à 90 secondes, les données répliquées sont considérées non synchronisées dans leur globalité. La réintégration est interrompue avec le message suivant dans le journal du module :

```
| 2021-08-06 08:40:20.909224 | reintegre | E | La synchronisation automatique ne peut être appliquée en raison d'un delta trop important entre les dates de dernière synchronisation
```

Si l'administrateur considère que le serveur est valide, il peut forcer le démarrage en secondaire avec synchronisation complète des données, en exécutant la commande : `safekit second fullsync -m AM`

5.5 Démarrage d'un module miroir avec les données à jour ✗ STOP (NotReady) - ○WAIT (NotReady)

SafeKit choisit quel serveur doit démarrer en tant que primaire. Pour cela, il retient le serveur avec les répertoires répliqués à jour. Pour profiter de cette fonctionnalité, utiliser la commande start et NON la commande `prim`.

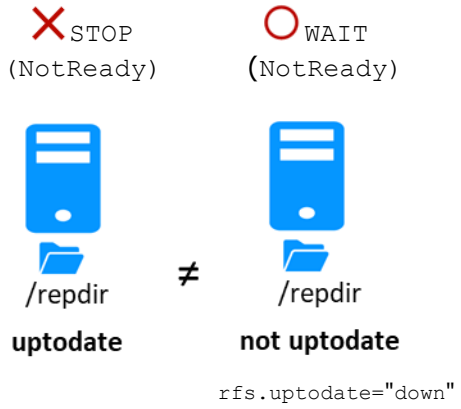
1. État initial • node1 est primaire `ALONE` • les répertoires sont à jour sur ce serveur • le module est arrêté sur node2 • node2 a des répertoires répliqués désynchronisés	
2. Commande `stop` sur node1 • arrêt du serveur avec les répertoires à jour	

3. Commande `start` sur node2

- le module est mis dans l'état `WAIT` en attendant le démarrage de l'autre serveur. Dans son journal de message :

  "Potentially not uptodate data for replicated directories (wait for the start of the remote server)"
  "Action wait from failover rule notuptodate_server"
  "If you are sure that this server has valid data, run safekit prim to force start as primary"

- dans ce cas, vous devez démarrer node1 pour resynchroniser node2
- si vous voulez réellement sacrifier les données à jour et démarrer node2 avec les données non à jour en tant que primaire : commande `stop` puis commande `prim` sur node2



Voir aussi la section 5.9

5.6 Mode de réplication dégradé (✓ALONE (Ready) dégradé)

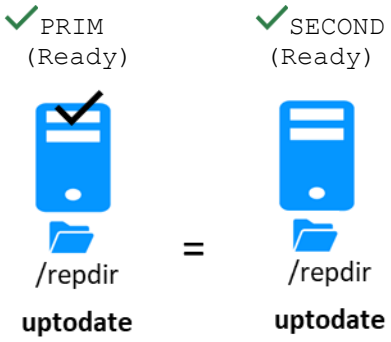
Si le processus de réplication `nfsbox` connait une défaillance sur la machine primaire (liée par exemple à une mauvaise configuration de la réplication de fichiers), l'application n'est pas basculée inutilement sur le serveur secondaire
Le serveur primaire va dans l'état `ALONE` et dans un mode de réplication dégradé. Cet état dégradé est affiché dans la console. Le message dans le journal est

"Resource rfs.degraded set to up by nfsadmin"

Et `safekit state -v -m` *AM* (où *AM* est le nom du module) présente la ressource `rfs.degraded` up
Le serveur primaire continue en `ALONE` avec un processus `nfsbox` qui ne réplique plus
Il faut arrêter et redémarrer le serveur `ALONE` pour revenir dans la situation `PRIM - SECOND` avec réplication

1. État initial

- le miroir est dans l'état stable node1 ✓`PRIM (Ready)` - serveur 2 ✓ `SECOND (Ready)`



2. Défaillance du processus de réplication `nfsbox` sur node1

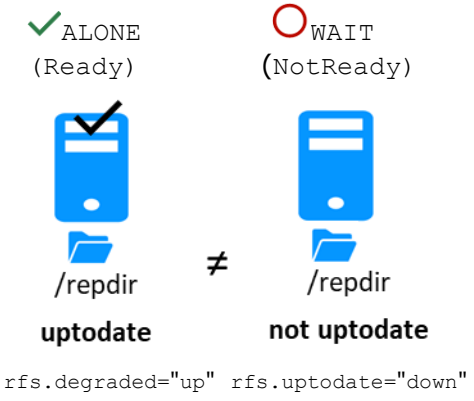
- node1 devient ✓`ALONE (Ready)` dégradé avec le message

  "set up of rfs.degraded called by nfsadmin"

  `safekit state -v` présente la ressource `rfs.degraded="up"`
- node1 continue à exécuter l'application sans réplication
- node2 se met dans l'état ○`WAIT (NotReady)` en attente du processus de réplication avec le message dans son journal

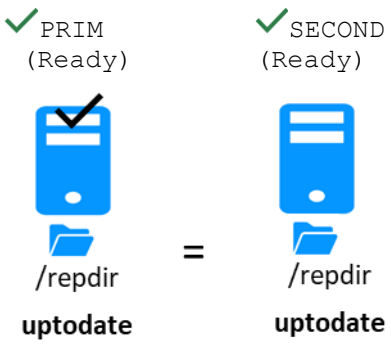
  "Action wait from failover rule degraded_server"

  et avec `rfs.uptodate="down"`



3. Retour à la réplication

- l'administrateur réalise `stop` ; `start` sur node1 `ALONE`
- le processus de réplication `nfsbox` est relancé sur node1
- node2 réintègre les répertoires répliqués avant de devenir ✓`SECOND (Ready)`
- node1 devient ✓`PRIM (Ready)`

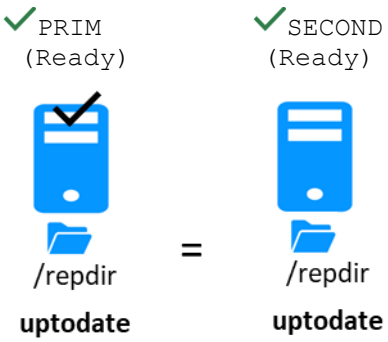


5.7 Reprise automatique ou manuelle `failover="off"` - ✕STOP (NotReady) - ○WAIT (NotReady)

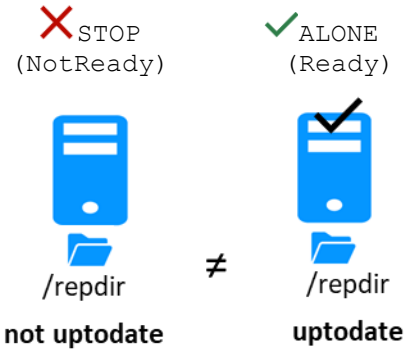
La reprise automatique ou manuelle sur le serveur secondaire est définie dans `userconfig.xml` par `<service mode="mirror" failover="on"|"off">`. Par défaut, si la valeur n'est pas définie, `failover="on"`

Le mode `failover="off"` est utile lorsque l'on veut contrôler le basculement par un administrateur. Ce mode assure qu'une application tourne toujours sur le même serveur primaire quel que soit les opérations sur ce serveur (reboot, arrêt temporaire du module pour maintenance...). Seule une commande d'un administrateur (commande `prim`) peut mettre l'autre serveur en primaire

1. **État initial**
   - le miroir est dans l'état stable node1 ✓`PRIM (Ready)` - serveur 2 ✓`SECOND (Ready)`

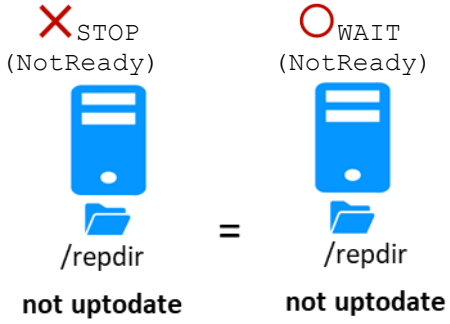


2. **Fonctionnement avec `failover="on"`**
   - si node1 anciennement `PRIM` rencontre une défaillance et s'arrête, node2 devient automatiquement primaire `ALONE` (mode par défaut)



3. **Fonctionnement avec `failover="off"`**
   - Si node1 anciennement `PRIM` connait une défaillance et s'arrête, node2 se met en ○ `WAIT (NotReady)` avec dans son journal le message

     "Failover-off configured"
     "Action stopstart called by failover-off"
     "Transition STOPSTART from failover-off"
     "Local state `WAIT NotReady` "
   - L'administrateur dans cette situation peut redémarrer node1 s'il n'est pas en panne : le miroir redémarre dans son ancien état serveur 1 ✓`PRIM (Ready)` - serveur 2 ✓ `SECOND (Ready)`
   - L'administrateur peut décider de forcer node2 à devenir primaire avec les commandes : `stop` ; `prim` sur node2

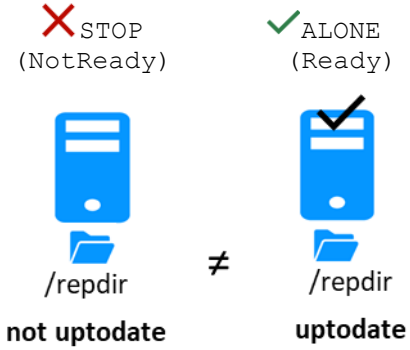
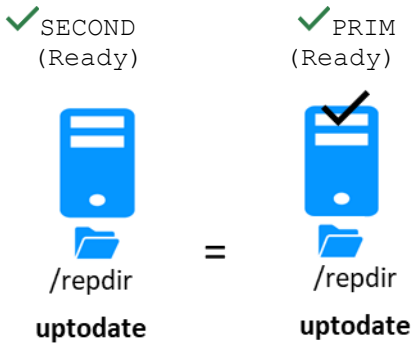


Voir aussi la section 5.9

5.8 Serveur primaire par défaut (swap automatique après réintégration)

A la réintégration après panne, un serveur redevient par défaut secondaire. L'administrateur peut choisir de ramener l'application sur le serveur réintégré à un moment opportun avec la commande `swap`. C'est le comportement par défaut lorsque dans `userconfig.xml` <service> est défini sans la variable `defaultprim`

Si l'on veut que l'application revienne automatiquement sur le serveur juste après sa réintégration, il faut configurer dans `userconfig.xml <service mode="mirror" defaultprim="hostname serveur 1">`

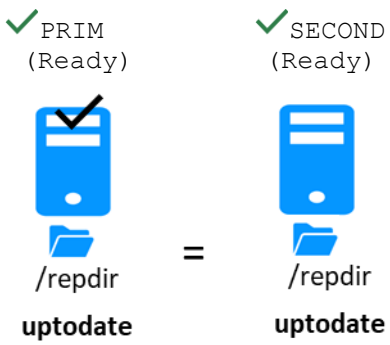
1. État initial • node1 (anciennement `PRIM`) connait une défaillance et s'arrête • node2 secondaire devient automatiquement primaire `ALONE`	
2. Retour sans `defaultprim` • node1 est relancé par la commande `start` : il réintègre les répertoires répliqués puis devient secondaire • un administrateur peut replacer le primaire sur node1 avec la commande `stopstart` du serveur 2à une heure propice • le stopstart provoque l'arrêt de l'application sur node2 et son redémarrage sur node1	

3. Retour avec `defaultprim="hostname serveur 1"`

- node1 ✗`STOP (NotReady)` du cas 1 (état initial) est relancé par start
- il réintègre les répertoires répliqués
- juste après la réintégration, un swap automatique est réalisé par node1 avec les messages dans son journal :

  "Transition SWAP from defaultprim"
  "Begin of Swap"

- l'application est alors automatiquement arrêtée sur node2 et relancée sur node1
- à la fin de l'opération, node1 est `PRIM`

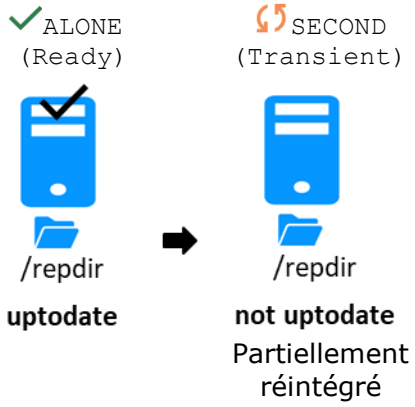


5.9 La commande `prim` échoue : pourquoi ? (commande `primforce`)

Il se peut qu'une commande `prim` échoue : après une tentative de démarrage, le serveur repasse en ✗`STOP (NotReady)`.

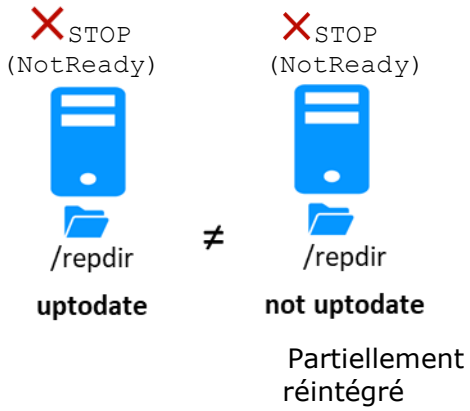
1. État initial

- node1 `ALONE` a les répertoires répliqués à jour
- node2 est en train de réintégrer les fichiers



2. `stop` sur node2 puis sur node1

- arrêt du serveur 2 pendant sa réintégration : l'arrêt du serveur 2 peut se faire alors qu'un fichier est à moitié recopié (fichier corrompu)
- node1 est lui aussi arrêté



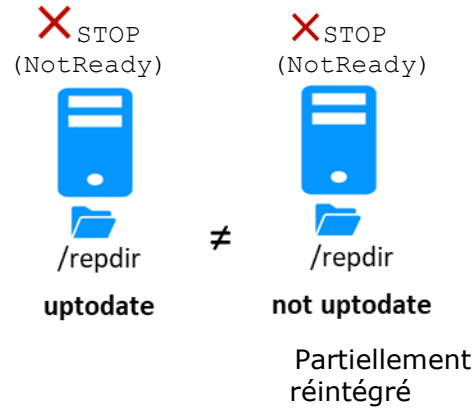
3. Commande `prim` sur node2

- la commande `prim` échoue : l'échec produit les messages suivants dans le log

  "Data may be inconsistent for replicated directories (stopped during reintegration)"
  "If you are sure that this server has valid data, run safekit primforce to force start as primary"

- dans ce cas, il faut démarrer node1 par la commande `start`. Et relancer node2 avec la commande `start` pour terminer la réintégration des fichiers. Tant que node2 n'a pas atteint l'état ✓ `SECOND (Ready)`, ses données ne sont pas intègres
- si vous voulez absolument démarrer sur node2 partiellement réintégré et avec des données potentiellement corrompues, utiliser la commande en ligne `safekit primforce -m` *AM* (où *AM* est le nom du module) sur node2. Message dans le journal :

  "Action primforce called by SYSTEM/root"



la commande `prim` échoue car les données peuvent être corrompues

Note : La commande `primforce` force une réintégration complète des répertoires répliqués sur la secondaire lorsqu'elle est démarrée.

6. Administration d'un module ferme

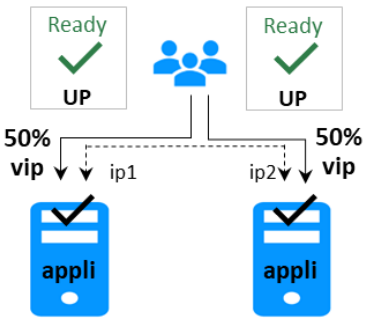
Pour tester un module ferme, voir la section 4.3.

Pour analyser un problème, voir la section 7.

6.1 Mode de fonctionnement d'un module ferme

1. Fonctionnement normal

État stable : 2 nœuds actifs.



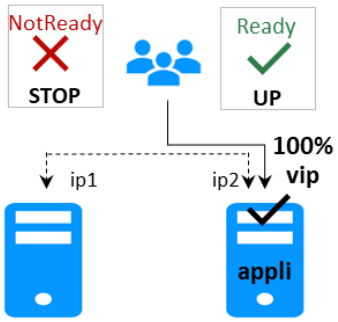
Sur tous les nœuds :

- IP virtuelle définie
- Application démarrée
- La charge du réseau est répartie entre tous les nœuds

Chaque nœud est prêt à effectuer une reprise automatique et assumer 100% de la charge.

2. Reprise automatique

État stable : 1 nœud actif.

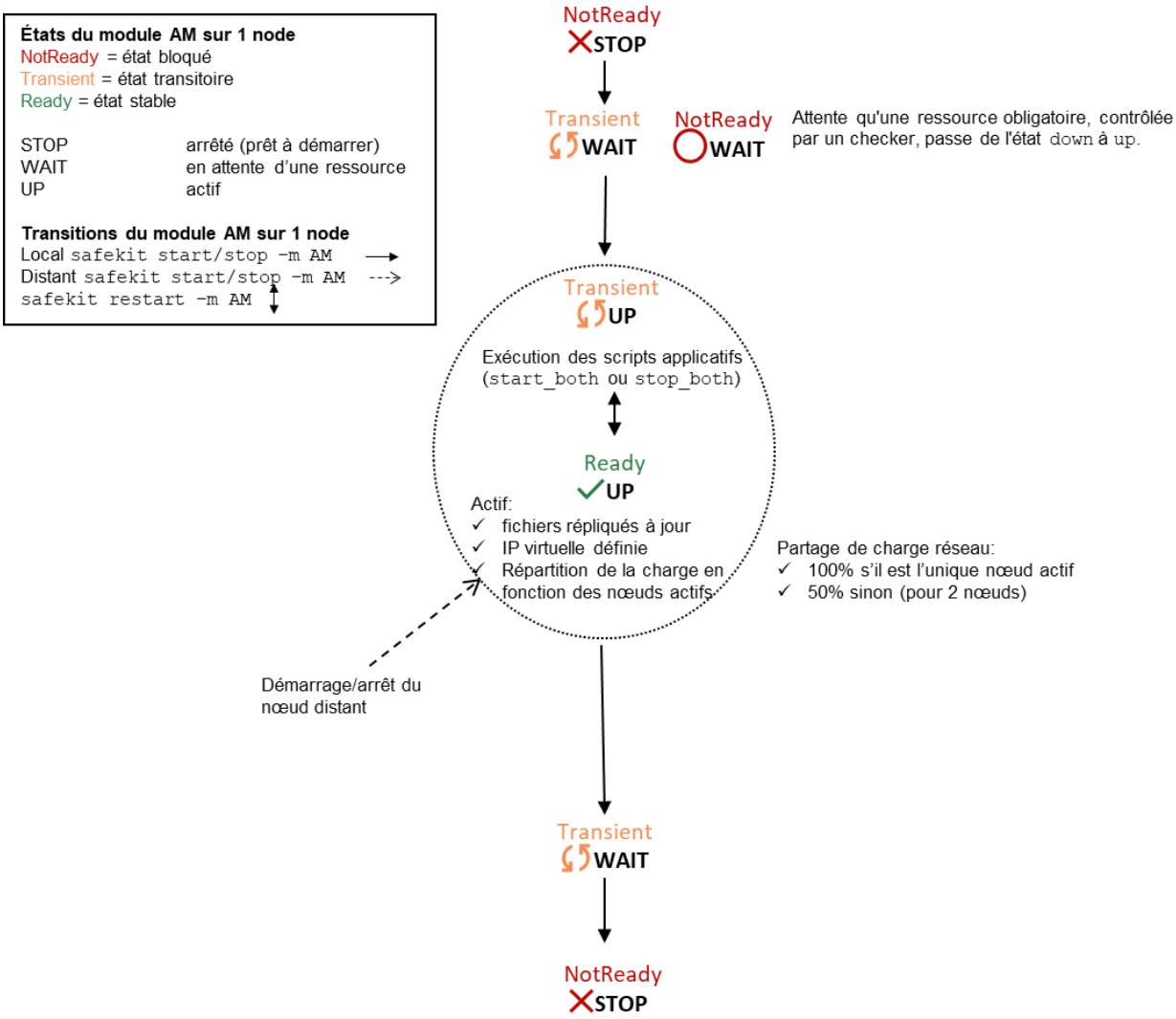


Sur arrêt du nœud distant, reprise automatique de toute la charge réseau.

3. Retour à la normale

État stable : 2 nœuds actifs.

6.2 Automate d'état d'un module ferme (STOP, WAIT, UP - NotReady, Transient, Ready)



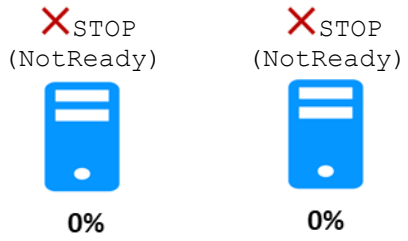
Note : C'est aussi l'automate d'un module de mode `light`. Ce mode se configure avec `<service mode="light">` dans le fichier `userconfig.xml` du module sous `SAFE/modules/`*`AM`*`/conf` (où *`AM`* est le nom du module). Le mode light correspond à un module s'exécutant sur un nœud sans synchronisation avec d'autres nœuds (comme peuvent le faire des modules miroir ou ferme). Un module light intègre les procédures de démarrage et d'arrêt d'une application ainsi que les checkers SafeKit qui permettent de détecter des erreurs.

6.3 Démarrage d'un module ferme

Il n'y a pas de procédure spéciale pour démarrer un module ferme : utiliser seulement la commande start sur tous les serveurs exécutant le module. Ci-dessous un exemple avec une ferme de 2 serveurs.

1. État initial

- le module ferme a été configuré sur 2 serveurs

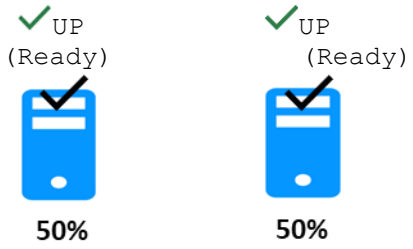


2. Commande `start` sur node1 et node2

- message dans le journal des 2 serveurs :

  "farm membership: **node1 node2** (group FarmProto_0)"
  "farm load: **128/256** (group FarmProto_0)"
  "Local state `UP` Ready"

- ressource de chaque instance du module sur les 2 serveurs : `FarmProto_0 50%`



7. Résolution de problèmes

7.1 Problème de connexion avec la console web

Si vous rencontrez des problèmes de connexion avec la console web, tels que pas de réponse du nœud ou erreur de connexion, appliquez les contrôles et procédures ci-dessous :

Ensuite, il peut être nécessaire de recharger la console dans le navigateur.

7.1.1 Contrôler le navigateur

Vérifiez pour le navigateur web :

1. que le navigateur et sa version sont bien supportés (dans certains environnements, Chrome fonctionne mieux qu'Internet Explorer)
2. modifiez le paramétrage du proxy pour définir une connexion directe ou indirecte au serveur

3. pour Internet Explorer, modifiez les paramètres de sécurité (ajoutez l'url dans les zones de sécurité)
4. sur évolution de version de SafeKit, nettoyez le cache du navigateur comme décrit plus loin
5. que la console web et le serveur ont la même version (la compatibilité peut ne pas être préservée)

7.1.2 Supprimer l'état du navigateur

Pour supprimer l'état du navigateur :

1. Videz son cache

   Ouvrir le navigateur sur n'importe quelle page web, et presser en même temps les touches Ctrl, Shift et Suppr. Cela ouvre une fenêtre de dialogue : cocher tous les items puis cliquez le bouton Nettoyer maintenant ou Supprimer

2. Videz le cache SSL si la console se connecte en HTTPS

   Dans les paramètres avancés du navigateur, rechercher le cache SSL et le vider

Fermez le navigateur, arrêtez tous les processus du navigateur qui continueraient à tourner en tâche de fond et relancez-le.

7.1.3 Contrôler les serveurs

Vérifiez sur chaque nœud du cluster SafeKit :

1. le pare-feu

   Si cela n'a pas encore été fait, exécutez la commande `SAFE/private/bin/firewallcfg add` qui configure le pare-feu du système d'exploitation. Pour les autres pares-feux, ajoutez des exceptions pour autoriser les connexions entre le navigateur web et le serveur. Pour les détails de configuration du pare-feu, voir la section 10.3.

2. la configuration du service web

   L'accès à la console web nécessite une authentification. Si cela n'a pas encore été fait, exécutez la commande `SAFE/private/bin/webservercfg -passwd` *pwd* pour initialiser (ou réinitialiser) cette configuration avec le mot de passe de l'utilisateur `admin`. Pour plus de détails, voir la section 11.2.1.

3. la disponibilité du réseau et du serveur
4. les services `safeadmin` et `safewebserver`

   Ils doivent être démarrés

5. la configuration du cluster

   Exécutez la commande `safekit cluster confinfo` (voir section 9.2). Elle doit retourner sur tous les nœuds, la même liste de nœuds et la même signature de configuration. Si ce n'est pas le cas, réappliquez la configuration du cluster sur tous les nœuds (voir section 12.2)

7.2 Problème de connexion HTTPS avec la console web

Si vous rencontrez des problèmes de connexion avec la console web en HTTPS, appliquez les contrôles et procédures ci-dessous :

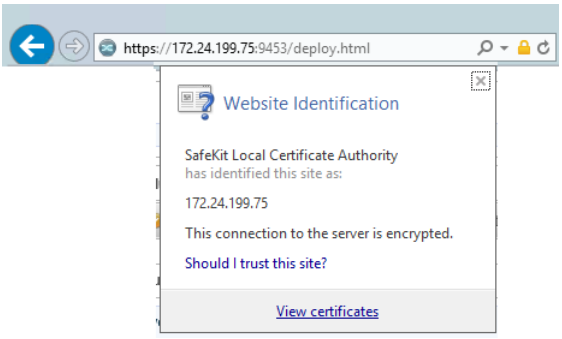
⇨ Section 7.1 « Problème de connexion avec la console web »

⇨ Section 7.2.1 « Contrôler les certificats serveurs »
⇨ Section 7.2.2 « Contrôler les certificats installés dans SafeKit »
⇨ Section 7.2.3 « Revenir à la configuration HTTP »

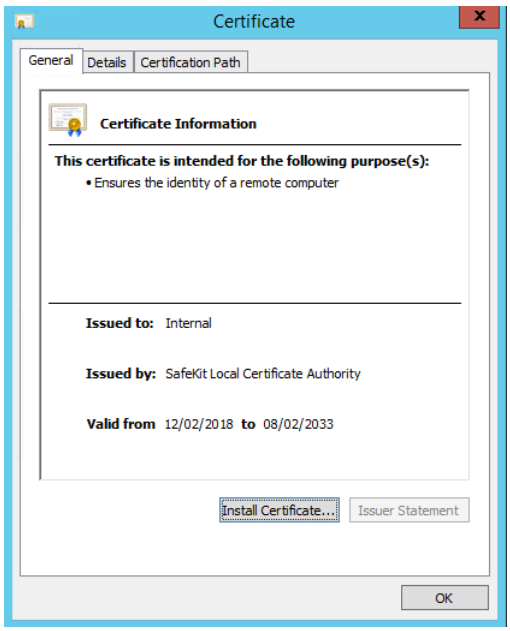
7.2.1 Contrôler les certificats serveurs

La console web se connecte à un nœud du cluster identifié par un certificat. Pour obtenir le contenu du certificat associé au nœud, exécutez les opérations suivantes si vous utilisez Edge ou Chrome :

1. Cliquez sur le verrou affiché à côté de l'URL pour ouvrir la fenêtre de sécurité
2. Cliquez sur le lien View certificates. Cela ouvre une fenêtre qui affiche le contenu du certificat

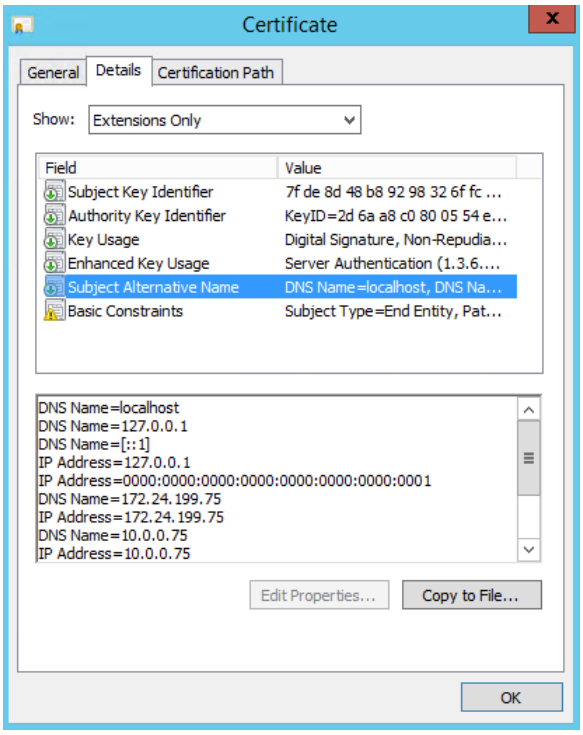


3. Vérifiez l'identité de l'émetteur qui doit être votre autorité de certification
4. Vérifiez la date de validité et la date de station de travail. Remettre la station à la bonne date si nécessaire
5. Vérifiez la date de validité. Si le certificat a expiré, vous devez le renouveler



6. Cliquez sur l'onglet Détails
7. Sélectionnez le champ Autre nom de l'objet. Son contenu est affiché dans le panneau inférieur. Le nom défini dans l'URL pour la connexion à la console web SafeKit doit être inclus dans cette liste. Changer l'URL si nécessaire
8. La valeur de l'attribut `adress` pour le serveur, telle que définie dans la configuration du cluster SafeKit, doit être incluse dans cette liste. Si ce n'est pas le cas, modifiez la configuration du cluster comme cela est décrit en section 12.2.

   Si vous utilisez le nom DNS, vous devez mettre le nom en minuscules.



Avec SafeKit <= 7.5.2.9, le nom du serveur doit être obligatoirement inclus.

7.2.2 Contrôler les certificats installés dans SafeKit

Vous pouvez utiliser la commande `checkcert` pour contrôler les certificats.

Sur chaque nœud du cluster SafeKit :

1. Se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
2. Aller dans le répertoire `SAFE/web/bin`
3. Exécuter `checkcert -t all`
4. La commande contrôle tous les certificats installés et échoue si une erreur est détectée
5. Exécuter la commande suivante pour vérifier que le certificat serveur contient bien un nom DNS ou une adresse IP donné :

   ```
   checkcert -h "DNS name value"
   ```

   ```
   checkcert -i "Numeric IP address value"
   ```

Le certificat de serveur doit contenir tous les noms DNS et/ou adresses IP utilisés pour la connexion HTTPS. Ceux-ci doivent également être inclus dans le fichier de configuration du cluster SafeKit.

7.2.3 Revenir à la configuration HTTP

Si le problème ne peut être résolu, vous pouvez revenir à la configuration HTTP. Sur tous les serveurs :

1. supprimer le fichier SAFE/web/conf/ssl/httpd.webconsolessl.conf
2. exécuter `safekit webserver restart`

   (SAFE=C:\safekit en Windows si System Drive=C: ; et SAFE=/opt/safekit en Linux):

3. vider le cache du navigateur comme décrit en section 7.1.2.

7.3 Comment lire les journaux et les ressources du module ?

Le **journal du module** et le **journal des scripts** sur un nœud peuvent être consultés avec (remplacer ci-dessous `node1` par le nom du nœud et *AM* par le nom du module) :

- la console web avec l'URI /console/fr/monitoring /modules/*AM*/nodes/node1/logs
- la commande `safekit logview -m AM` exécutée sur `node1`, pour le journal du module
- sur `node1`, dans les fichiers `SAFEVAR/modules/AM/userlog_<year>_<month>_<day>T<time>_<script name>.ulog`, pour le journal des scripts

Avec le journal du module, vous pouvez comprendre pourquoi un module n'est plus dans état stable ✓`(Ready)`.

Avec le journal des scripts, vous aurez l'output des scripts utilisateur (`start_xxx` et `stop_xxx`).

Noter qu'un module peut quitter son état stable ✓`(Ready)` à cause d'une commande administrateur : `safekit start | stop | restart | swap | stopstart | forcestop`

- Vous trouverez une liste des messages du journal en index
- Les messages dans le journal après une commande administrateur sont :

```
"Action start called by admin@<IP>/SYSTEM/root"
"Action stop called by admin@<IP>/SYSTEM/root"
"Action restart called by admin@<IP>/SYSTEM/root"
"Action swap called by admin@<IP>/SYSTEM/root"
"Action stopstart called by admin@<IP>/SYSTEM/root"
"Action forcestop called by admin@<IP>/SYSTEM/root"
```

admin@<ip>: via la console
SYSTEM: ligne de commande Windows
root: ligne de commande Linux

- Si "Action stop called by maxloop" apparaît dans le journal du module, voir la section 7.11

L'état des **ressources** du module sur un nœud peut être analysé avec (remplacer ci-dessous `node1` par le nom du nœud et *AM* par le nom du module) :

- la console web avec l'URI /console/fr/monitoring/modules/*AM*/nodes/node1/resources
- la commande `safekit state -m AM -v` exécutée sur `node1`

- status du module

```
state.local, state.remote
usersetting.errd, usersetting.checker,
usersetting.encryption
```

- Checkers

```
proc.xxx, intf.xxx, custom.xxx
```

- Réplication de fichiers

```
rfs.uptodate, rfs.degraded,
rfs.reintegre_failed
```

7.4 Comment lire le journal de commandes du serveur ?

Il existe un journal des commandes exécutées sur le serveur Safekit.

Le **journal des commandes** peut être consulté avec la commande `safekit cmdlog`

Pour plus de détails, voir la section 10.10.

7.5 Module stable ✓(Ready) et ✓(Ready)

- Un module miroir stable sur 2 serveurs est dans l'état ✓PRIM (Ready) - ✓SECOND (Ready) : l'application est opérationnelle sur le serveur PRIM ; en cas de panne, le serveur SECOND est prêt à reprendre l'application.
- Un module ferme stable est dans l'état ✓UP (Ready) sur tous les serveurs de la ferme : l'application est opérationnelle sur tous les serveurs.

7.6 Module dégradé ✓(Ready) et ✗/○(NotReady)

Un module miroir dégradé est dans l'état ✓ALONE (Ready) - ✗STOP/○WAIT (NotReady). Il n'y a plus de serveur de reprise mais l'application est opérationnelle sur le serveur ALONE.

Un module ferme dégradé est dans l'état ✓UP (Ready) sur au moins un serveur de la ferme, les autres serveurs étant dans l'état ✗STOP/○WAIT (NotReady). L'application est opérationnelle sur le serveur UP.

Dans le cas dégradé, il n'y a pas de procédure d'urgence à mettre en œuvre. L'analyse de l'état ✗STOP/○WAIT (NotReady) peut être réalisée plus tard. Néanmoins, vous pouvez tenter de redémarrer le module :

⇨ si le module est ✗STOP, voir la section 7.8

⇨ si le module est ✗WAIT, voir la section 7.9

7.7 Module hors service ✗/○ (NotReady) et ✗/○ (NotReady)

Un module miroir ou ferme hors service est dans l'état ✗STOP/○WAIT (NotReady) sur tous les serveurs. Dans ce cas, l'application n'est plus opérationnelle sur aucun serveur. Il faut rétablir la situation et redémarrer le module dans l'état ✓(Ready) sur au moins un serveur :

⇨ si le module est ✗STOP (NotReady), voir la section 7.8

⇨ si le module est ○WAIT (NotReady), voir la section 7.9

7.8 Module ✗STOP (NotReady) : redémarrer le module

1. Redémarrer le module arrêté (remplacer ci-dessous *AM* par le nom du module) avec :
   - La console web via 👁 Supervision/••• du nœud/▷ Démarrer/
   - la commande `safekit start -m AM` exécutée sur le nœud
2. Vérifier que le module devient ✓(Ready).
3. Analyser le résultat du démarrage dans le journal du module et le journal des scripts avec (remplacer ci-dessous `node1` par le nom du nœud et *AM* par le nom du module) :
   - la console web avec l'URI /console/fr/monitoring /modules/*AM*/nodes/node1/logs
   - la commande `safekit logview -m AM` exécutée sur `node1`, pour le journal du module

- sur node1, dans les fichiers SAFEVAR/modules/*AM*/userlog_<year>_<month>_<day>T<time>_<script name>.ulog, pour le journal des scripts

7.9 Module ○WAIT (NotReady) : réparer la ressource="down"

Si le module est dans l'état ○WAIT (NotReady), il attend que l'état d'une ressource devienne up.

Vous devez réparer la ressource mise à down.

Pour déterminer la ressource à réparer, analyser les messages du journal et l'état des ressources (voir section 7.3).

Notes :

Un checker de type wait est à l'origine de l'état ○WAIT (NotReady). Il est démarré après le script prestart et arrêté avant poststop.

Le checker est actif sur tous les serveurs ✓ALONE/PRIM/SECOND/UP (Ready).

L'action du checker sur erreur est de positionner une ressource à down.

La règle de failover associée à la ressource down exécute l'action wait.

Le module est mis localement dans l'état ○WAIT (NotReady) tant la ressource reste down.

Le module sort de l'état ○WAIT (NotReady) dès que le checker positionne la ressource à up.

Messages des checkers wait :

- fichiers non à jour localement : voir section 5

  "Potentially not uptodate data for replicated directories (wait for the start of the remote server)"
  "Action wait from failover rule notuptodate_server"
  "If you are sure that this server has valid data, run safekit prim to force start as primary"

- <interface check="on"> checker d'une interface réseau locale

  "Resource intf.ip.0 set to down by intfcheck"
  "Action wait from failover rule interface_failure"

- <ping> checker d'une adresse IP externe

  "Resource ping.id set to down by pingcheck"
  "Action wait from failover rule ping_failure"

- <module> checker d'un autre module

  "Resource module.othermodule_ip set to down by modulecheck"
  "Action wait from failover rule module_failure"

- <tcp ident="id" when="pre"> checker d'un service TCP externe

  "Resource tcp.id set to down by tcpcheck"
  "Action wait from failover rule t_id"

- <custom ident="id" when="pre"> checker customisé

  "Resource custom.id set to down by customscript"
  "Action wait from failover rule customid_failure"

- <splitbrain> checker

  "Resource splitbrain.uptodate set to down by splitbraincheck"
  "Action wait from failover rule splitbrain_failure"

	Fichiers non à jour localement à cause du split-brain : voir section 13.17

7.10 Module oscillant de ✓ (Ready) à ↻ (Transient)

Si un module oscille de l'état ✓ (Ready) à l'état ↻ (Transient), il est soumis à un checker de type restart ou stopstart qui détecte une erreur en boucle. Par défaut, au 4ième redémarrage infructueux sur un serveur, le module s'arrête sur le serveur en ✗ STOP (NotReady). Analyser le journal du module pour déterminer quel checker est à l'origine de l'oscillation (pour lire les journaux, voir la section 7.3). **Notes :** Un checker restart ou stopstart est défini dans userconfig.xml par : • when="prim" pour un module miroir Le checker est démarré sur le nœud ✓ PRIM/ALONE (Ready) après le script start_prim (stoppé avant stop_prim). Il teste l'application démarrée dans start_prim. • when="both" pour un module ferme Le checker est démarré sur tous les nœuds ✓ UP (Ready) après le script start_both (stoppé avant stop_both). Il teste l'application démarrée dans start_both. L'action du checker sur erreur est d'exécuter un restart ou stopstart du module. stopstart sur ✓ PRIM (Ready) amène à une reprise du rôle de primaire sur l'autre nœud. Le module est dans l'état ↻ PRIM/UP (Transient) pendant la phase de redémarrage Après plusieurs oscillations, le module s'arrête avec le message "Action stop called by maxloop" dans le journal du module : voir section 7.11	Messages des checkers restart ou stopstart : • <errd> dans userconfig.xml: checker de processus "Process appli.exe not running" "Action restart|stopstart called by errd" • <tcp ident="id" when="prim"|"both"> dans userconfig.xml: checker TCP d'une application "Resource tcp.id set to down by tcpcheck" "Action restart|stopstart from failover rule tcp_failure" • <custom ident="id" when="prim"|"both"> dans userconfig.xml : checker customisé "Resource custom.id set to down by customscript" "Action restart|stopstart from failover rule customid_failure" ou "Action restart|stopstart called by customscript"

7.11 Message sur stop après `maxloop`

Si une erreur détectée par un checker se répète plusieurs fois et successivement, le module est arrêté sur le serveur en ✗`STOP (NotReady)` car l'erreur est permanente et l'action du checker n'arrive pas à la corriger

Si dans `userconfig.xml`, pas de paramètre `maxloop` / `loop_interval` dans `<service>` :

- par défaut `maxloop="3"`, `loop_interval="24"`
- si les checkers génèrent plus de 3 redémarrages infructueux (`restart`, `stopstart`, `wait`) en moins de 24H, alors stop du module : ✗`STOP (NotReady)`

Le compteur est remis à 0 dès lors qu'une action de type administrateur est réalisée sur le module : comme une commande start ou stop

Message sur stop après `maxloop`

"Action stop called by maxloop"

7.12 Module ✓`(Ready)` mais application non opérationnelle

Si un serveur présente un état ✓`PRIM (Ready)` ou ✓`ALONE (Ready)` ou ✓`UP (Ready)`, il se peut que l'application soit non opérationnelle à cause d'erreurs au démarrage non détectées. Dans la suite, remplacer `node1` par le nom du nœud et *`AM`* par le nom du module.

1. Analyser les messages de sortie de l'application produits par `start_prim`(/`start_both`) et `stop_prim`(/`stop_both`). Ils sont visibles avec :
   - la console web avec l'URI /console/fr/ monitoring /modules/*AM*/nodes/node1/logs
   - sur node1, dans les fichiers SAFEVAR/modules/*AM*/userlog_<year>_<month>_<day>T<time>_<script name>.ulog, pour le journal des scripts

   Chercher s'il y a des erreurs dans les phases de démarrage/arrêt de l'application. Attention, parfois le journal des scripts est désactivé car trop volumineux avec `<user logging="none">` dans `userconfig.xml` du module.
2. Vérifier les scripts `start_prim`(/`start_both`) et `stop_prim`(/`stop_both`) du module miroir(/ferme) et `userconfig.xml` avec :
   - la console web avec l'URI /console/fr/configuration/modules/*AM*/config
   - sur `node1` dans le répertoire `SAFE/modules/`*`AM`*
3. Faire un `restart` du le nœud ✓`PRIM/ALONE/UP (Ready)` pour arrêter et redémarrer localement l'application (sans basculement) avec :

- la console web via Supervision/••• du nœud/Redémarrer/
- la commande `safekit restart -m AM` exécutée sur le nœud

4. Si l'application est toujours non opérationnelle, appliquer un `stop` sur le nœud `PRIM/ALONE/UP (Ready)` pour arrêter le module et l'application (avec basculement sur l'autre nœud si ce dernier est `Ready`) :
   - la console web via Supervision/••• du nœud/ Arrêter/
   - la commande `safekit stop -m AM` exécutée sur le nœud

7.13 Module mirror ALONE (Ready) / WAIT ou STOP (NotReady)

Si un module miroir reste dans l'état `ALONE (Ready)` / `WAIT (NotReady)`, vérifier la ressource `state.remote` sur chacun des nœuds (pour lire les ressources, voir la section 7.3). Si cet état est UNKNOWN sur les deux nœuds, alors il s'agit probablement d'un problème de communication entre nœuds. Cette situation peut aussi amener à l'état `ALONE (Ready)` / `STOP (NotReady)`. Les raisons possibles sont :

1. Problème réseau

   Vérifier la configuration réseau

2. Règles de pare-feu sur l'un ou les deux nœuds

   Voir la section 10.3

3. Configuration du cluster ou clés cryptographiques du cluster non identiques

   Afin de communiquer entre eux, les nœuds doivent appartenir au même cluster SafeKit et avoir la même configuration (voir section 12).

   - La console web émet un message d'avertissement si les nœuds du cluster n'ont pas la même configuration
   - La commande en ligne : `safekit cluster confinfo` exécutée sur n'importe quel nœud du cluster doit reporter des signatures de configuration de cluster identiques pour tous les nœuds (voir section 9.2)

   Si la configuration du cluster SafeKit n'est pas identique, il faut réappliquer la configuration sur tous les nœuds comme cela est décrit en section 3.2.2.

4. Clés cryptographiques de module non identiques

   Quand la cryptographie est activée pour le module, la resource `usersetting.encryption` est "`on`" nœuds (pour lire les ressources, voir la section 7.3). Si les nœuds ont des clés cryptographiques différentes, alors les deux nœuds ne pourront pas communiquer entre eux.

   Afin de distribuer des clés identiques, il faut réappliquer la configuration du module sur tous les nœuds.

   Pour plus de détails, voir la section 10.6

5. Clés cryptographiques du module expirées

   Dans SafeKit <= 7.4.0.31, la clé de chiffrement des communications a une durée de validité de 1 an. Quand celle-ci expire dans un module miroir avec la réplication de fichiers, la réintégration sur le secondaire échoue et le module s'arrête avec le message d'erreur suivant dans le journal :

```
reintegre | D | XXX clnttcp_create: socket=7 TLS handshake failed
```

Dans SafeKit > 7.4.0.31, le message est :

```
reintegre | D | XXX clnttcp_create: socket=7 TLS handshake failed. Check server time and module certificate (expiration date, hash)
```

Pour résoudre ce problème, voir la section 10.6.3.1

7.14 Module ferme ✓UP (Ready) mais problème de load balancing

Bien que tous les serveurs de la ferme soient ✓UP (Ready), le load-balancing ne fonctionne pas.

7.14.1 Non cohérence des parts de la charge réseau

Dans un module ferme, la somme des parts de la charge réseau des nœuds ✓UP (Ready) doit être égale à 100%.

Si ce n'est pas le cas, il est très probable qu'il s'agisse d'un problème de communication entre nœuds. Les causes probables sont les mêmes que pour un module miroir, aussi voir la section 7.13 pour d'éventuelles solutions.

Voir aussi la section 4.3.6.

7.14.2 L'adresse IP virtuelle ne répond pas correctement

Si l'adresse IP virtuelle ne répond pas correctement à toutes les demandes de connexions :

1. choisir un nœud de la ferme qui reçoit et traite des connexions sur l'adresse IP virtuelle (connexions TCP établies) :
   - en Windows, utiliser la commande netstat -an | findstr <adresse IP virtuelle>
   - en Linux, utiliser la commande netstat -an | grep <adresse IP virtuelle>
2. arrêter le module ferme sur tous les nœuds excepté celui qui reçoit des connexions et qui doit rester ✓UP (Ready) avec :
   - la console web via Supervision/••• du nœud/ ☐ Arrêter/
   - la commande `safekit stop -m AM` sur les nœuds devant être arrêtés (où *AM* est le nom du module)
3. vérifier que l'ensemble des connexions vers l'adresse IP virtuelle sont traitées par le seul nœud ✓UP (Ready)

Pour une analyse plus fine sur ce sujet, voir :

⇨ section 4.3.4 pour le test de l'adresse virtuelle

⇨ section 4.3.5 pour le test du load-balancing

⇨ section 4.3.7 dans le cas d'une adresse MAC invisible

7.15 Problème après boot

Si vous rencontrez un problème après le boot, voir la section 4.1.

Notez que par défaut, les modules ne sont pas automatiquement démarrés au boot. Pour cela, vous devez configurer le démarrage au boot :

- avec la console web avec l'URI /console/fr/configuration/modules/*AM*/config

- dans le fichier `SAFE/modules/AM/conf/userconfig.xml` sur `node1` avec l'attribut `boot` dans le tag `service` (voir section 13.2.3)

Puis appliquer la nouvelle configuration sur tous les nœuds.

7.16 Analyse à partir des snapshots du module

Lorsque le problème n'est pas facilement identifiable, il est recommandé de prendre un snapshot du module sur tous les nœuds comme décrit dans la section 3.5. Un snapshot est un fichier zip qui rassemble, pour un module, les fichiers de configuration, les dumps… Son contenu permet une analyse hors ligne et approfondie de l'état du module et du nœud.

La structure et le contenu du snapshot varie en fonction de la version de SafeKit.

Depuis SafeKit 8.1, la structure du snapshot est la suivante :

snapshot_centos7-test3_mirror	`snapshot_`*`nodename`*`_`*`AM`* Snapshot pour le module *AM* récupéré depuis le nœud *nodename*
mirror	*AM* Nom du module
config_2021_05_05_14_15_42 config_2021_07_08_16_34_05 config_2021_08_05_16_35_08	`config_year_month_day_hour_mn_sec` Les 3 dernières configurations du module, y compris la courante
dump_2021_05_06_09_10_40 dump_2021_07_16_19_18_03 dump_2021_08_06_09_18_46	`dump_year_month_day_hour_mn_sec` les 3 derniers dump du module, y compris le dernier
tmp	pour le support niveau 3

7.16.1 Fichiers de configuration du module

Les fichiers de configuration du module sont sauvegardés comme suit :

config_2021_08_05_16_35_08 module bin conf web private	Le répertoire `module` contient les fichiers de configuration de l'utilisateur : • Répertoire `bin` scripts `start_xx`, `stop_xx`, … • Répertoire `conf` Fichier de configuration XML `userconfig.xml`

Vérifier le fichier de configuration XML et les scripts pour résoudre les problèmes d'intégration de l'application dans SafeKit

7.16.2 Fichiers de dump du module

Le dump contient l'état du module et du nœud SafeKit tel qu'il était au moment du dump.

csv licenses userlog var web	• Répertoire `csv` Journaux et états dans le format csv • Répertoire `licences` Licences SafeKit sauvegardées depuis le répertoire `SAFE/conf` • Répertoire `userlog` Journaux des scripts • Répertoire `var` Copie d'une partie du répertoire `SAFEVAR` • Répertoire `web` Fichiers de configuration du service web, copiés depuis le répertoire `SAFE/web/conf`
log.txt logverbose.txt	• Journaux du module (verbeux et non verbeux)
heartplug	• Fichier d'informations Diverses informations sur le nœud (liste et état des modules installés, version du système d'exploitation, configuration des disques et du réseau, etc.)
last.txt systemevt.txt ou systemevt.txt applicationevt.txt	• Journaux système En Linux, `last.txt` et `systemevt.txt` ou En Windows, `applicationevt.txt` et `systemevt.txt`
commandlog.txt	• Journal des commandes du nœud
heart heart.trc nfsbox nfsbox.trc	• Fichiers de trace pour le support niveau 3

- Vérifier le(s) fichier(s) de licence dans le répertoire `licenses` pour résoudre des problèmes concernant le contrôle de licence SafeKit
- Vérifier les fichiers de configuration Apache dans le répertoire `web` pour résoudre des problèmes concernant le service web SafeKit

- Vérifiez les logs du module, `log.txt` et `logverbose.txt`, pour résoudre des problèmes concernant le comportement du module
- Vérifier le journal des scripts userlog/`userlog_<year>_<month>_<day>T<time>_<script name>.ulog`pour résoudre des problèmes concernant le démarrage/arrêt de l'application
- Si nécessaire, consulter le fichier `heartplug` pour obtenir des informations sur le nœud et rechercher dans les journaux du système les événements qui se sont produits en même temps que le problème analysé
- Consulter le journal des commandes `commandlog.txt` pour résoudre des problèmes concernant la gestion du cluster SafeKit ou les commandes distribuées

7.16.2.1 Répertoire `var`

Le répertoire `var` est principalement destiné au support de niveau 3. Il s'agit d'une copie d'une partie du répertoire `SAFEVAR`. Dans le répertoire `var/cluster` :

- Consulter le fichier `cluster.xml` pour vérifier la configuration du cluster
- Consulter le fichier `cluster_ip.xml` pour résolution des noms DNS contenus dans la configuration du cluster

7.16.2.2 Répertoire `csv`

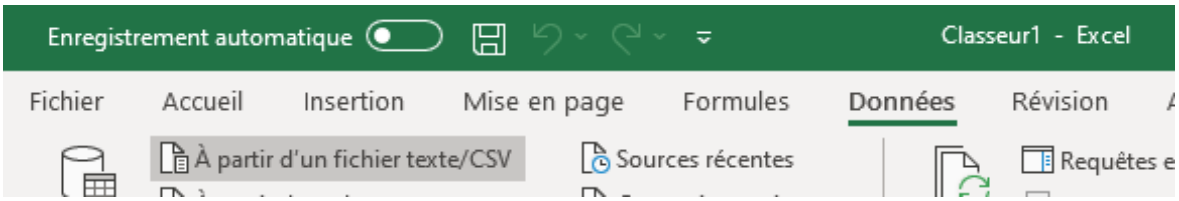
Les journaux et états sont aussi exportés au format csv, dans le répertoire `csv` :

csv	
logverbose.csv resource.csv resourcelog.csv	• Journaux et états du module Journal verbeux Etat des ressources Historique de l'état des ressources
commandlog.csv modules.csv moduleslog.csv clusterstate.csv	• Journaux et états du nœud Journal des commandes Liste des modules installés Pour le support niveau 3 Pour le support niveau 3

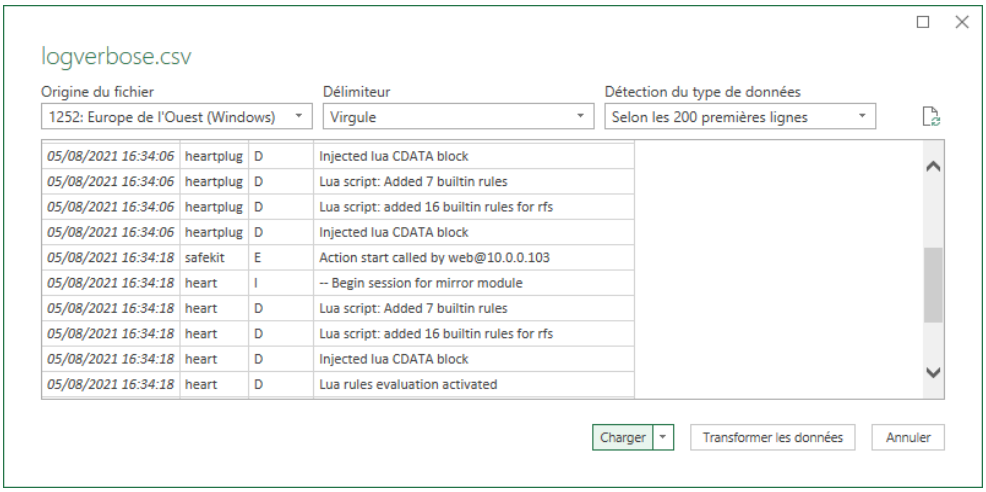
- Importer les fichiers csv dans Excel pour simplifier leur analyse

  Pour importer un fichier :

  - Créer un nouveau classeur
  - Depuis l'onglet Données, importer A partir d'un fichier text/CSV



- Dans la boîte de dialogue, localiser et double-cliquer sur le fichier csv à importer, puis cliquez sur Importer
- Puis cliquer sur Charger



Vous pouvez utiliser les fonctionnalités d'Excel pour filtrer les lignes en fonction du niveau des messages, … et charger dans des feuilles différentes les csv de chaque nœud.

Pour afficher la date exacte, formater les cellules avec `Nombre/Personnalisée jj/mm/aaaa hh:mm:ss,000`.

7.17 Problème avec la taille des bases de données de SafeKit

SafeKit utilise le stockage SQLite3 pour sauvegarder :

- Le journal et l'état du nœud
- `SAFEVAR/log.db` contient le journal des commandes
- `SAFEVAR/resource.db` contient la liste des modules installés et son historique

  Ces bases sont appelées bases de données du nœud.
- Le journal et les ressources du module
  - `SAFEUSERVAR/log.db` contient le journal du module.
  - `SAFEUSERVAR/resource.db` contient l'état des ressources du module et son historique

  Ces bases sont appelées bases de données du module.

La taille des logs et des historiques augmente au fur et à mesure que des événements se produisent sur le nœud SafeKit et les modules. Par conséquent, ils doivent être purgés régulièrement en supprimant les entrées les plus anciennes. Ceci est fait automatiquement grâce à une tâche périodique (task cheduler sous Windows ; crontab sous Linux) qui est contrôlé par le service `safeadmin`. Le nettoyage des bases de données du nœud est toujours actif. Le nettoyage des bases de données du module n'est

actif que lorsque le module est en cours d'exécution. Pour vérifier que les tâches sont actives :

- Tâche de nettoyage des bases de données du nœud
  - Sous Windows, exécutez schtasks /QUERY /TN safelog_clean
  - Sous Linux, exécutez crontab -u safekit -l

    La sortie de cette commande doit contenir l'entrée `safelog_clean`
- Tâche de nettoyage des bases de données du module *AM* (où *AM* est le nom du module)
  - Sous Windows, exécutez schtasks /QUERY /TN safelog_*AM*
  - Sous Linux, exécutez crontab -u safekit -l

    La sortie de cette commande doit contenir l'entrée `safelog_clean_AM`

La commande qui implémente le nettoyage est localisée sous `SAFEBIN` (en Linux, `SAFEBIN=/opt/safekit/private/bin`; en Windows, `SAFE=C:\safekit\private\bin` - si `%SYSTEMDRIVE%`=C:):

`dbclean.ps1` en Windows et `dbclean.sh` en Linux	Purge du journal et de l'historique dans les bases de données du nœud
`dbclean.ps1` *AM* en Windows et `dbclean.sh` *AM* en Linux	Purge du journal et de l'historique dans les bases de données du module nommé *AM*

Si nécessaire, vous pouvez exécuter ce script en dehors de la période prévue pour forcer le nettoyage des bases de données.

7.18 Problème pour récupérer le certificat de l'autorité de certification depuis une PKI externe

Lorsque vous utilisez la PKI SafeKit, vous devez fournir le certificat de l'autorité de certification `CA` utilisée pour émettre les certificats des serveurs (fichier `cacert.crt` contenant la chaîne de certificats pour les autorités de certification racine et intermédiaires).

Si vous rencontrez des difficultés à récupérer ce fichier depuis la PKI, vous pouvez le construire en suivant les procédures décrites ci-dessous.

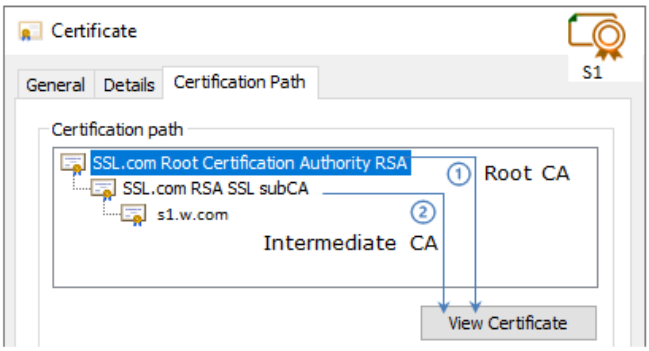
7.18.1 Exporter les certificats CA depuis des certificats publics

La procédure suivante explique comment construire, le fichier `combined.cer`, la chaîne de certificats pour les Autorités de Certification racine et intermédiaires d'un certificat public.

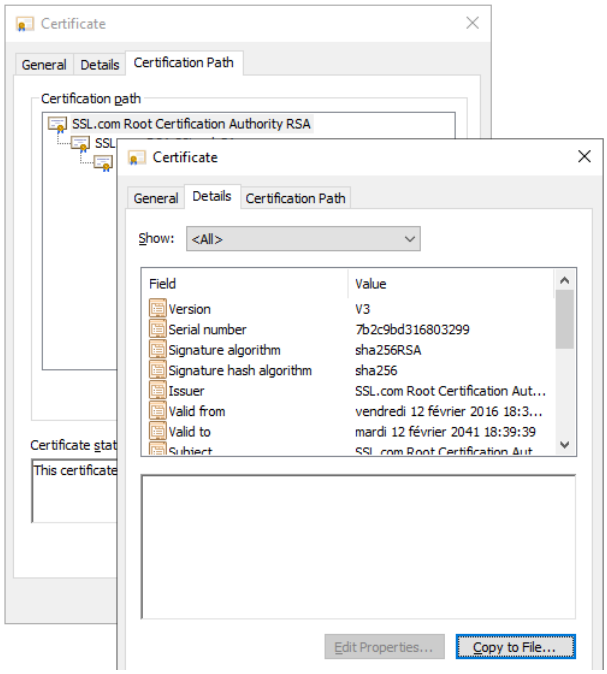
Lorsque vous avez le certificat public d'un serveur (fichier .crt ou .cer au format X.509 encodé en base-64) généré par la PKI :

1. Copier le fichier .crt (ou .cer) sur une station de travail Windows
2. Double cliquer sur le fichier pour l'ouvrir avec « Extension noyau de chiffrement »

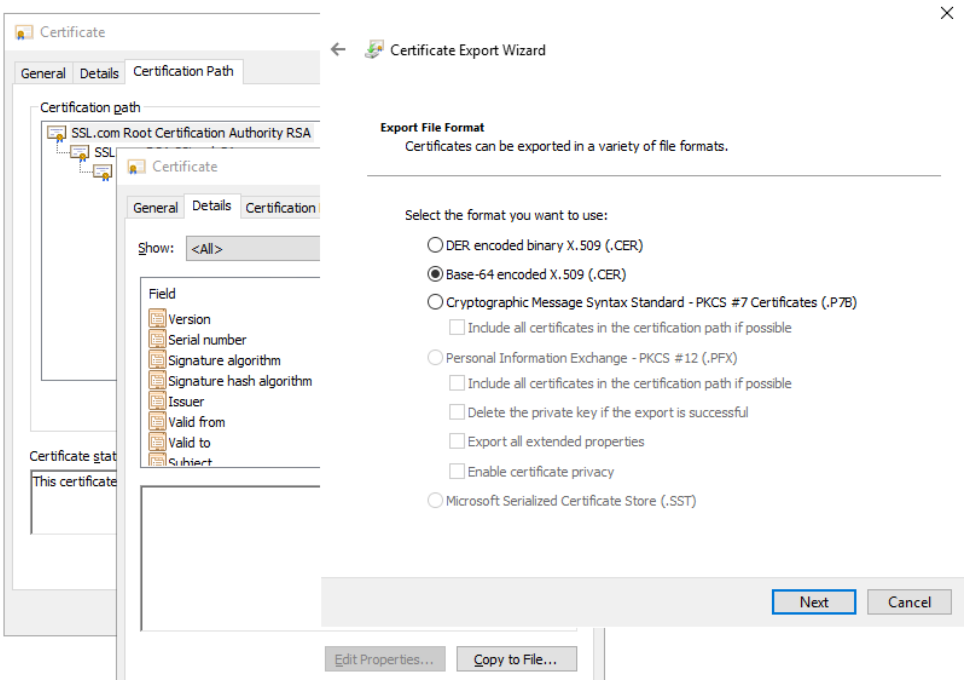
3. Cliquer sur l'onglet « Chemin d'accès de certification » pour afficher l'arbre des autorités de certification
4. Sélectionner une entrée (de haut en bas en excluant la feuille)



5. Cliquer sur « Afficher le certificat ». Une nouvelle fenêtre s'ouvre pour le certificat sélectionné
6. Dans cette nouvelle fenêtre, sélectionner l'onglet « Details » puis cliquer sur « Copier dans un fichier »

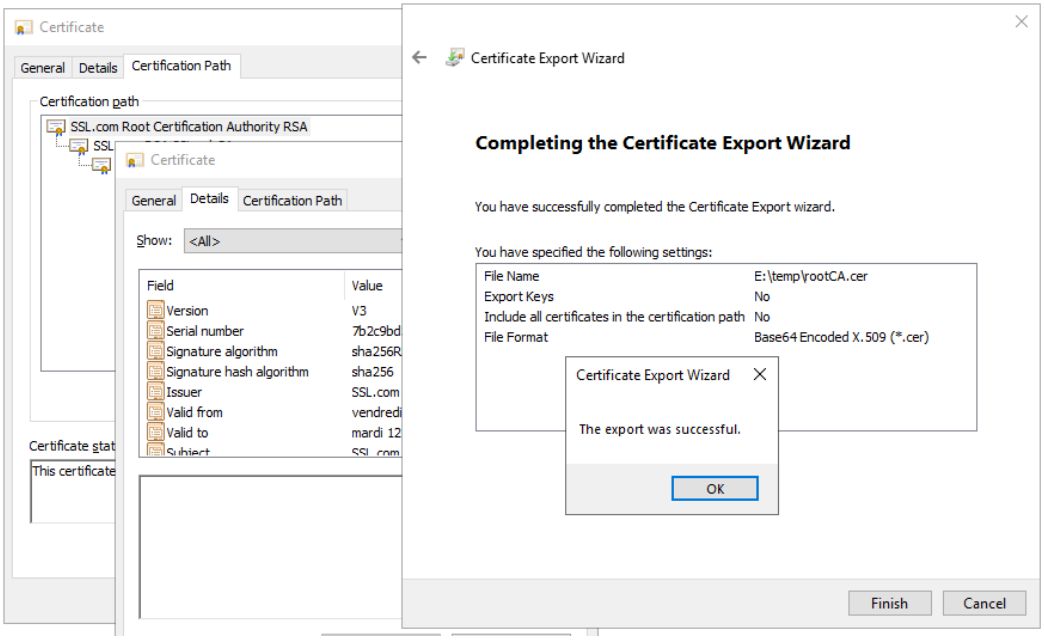


7. Cela ouvre l'Assistant Exportation du certificat :
    a. Cliquer sur Suivant
    b. Sur la page « Format de fichier d'exportation », sélectionner « Codé à base 64 X.509 (.cer). », puis cliquer sur « Suivant »



c. Dans « Fichier à exporter », cliquer sur « Parcourir » pour accéder à l'emplacement vers lequel vous souhaitez exporter le certificat. Pour la zone « Nom de fichier », nommer le fichier de certificat. Cliquer ensuite sur « Suivant ».

d. Cliquer sur « Terminer » pour exporter le certificat



8. Répéter maintenant les étapes 4 à 7 pour toutes les entrées (sauf la dernière) afin d'exporter tous les certificats des CA intermédiaires. Dans l'exemple, il faut répéter les étapes sur l'AC intermédiaire SSSL.com RSA subCA pour l'extraire en tant que certificat propre.
9. Concaténer tous les certificats obtenus dans un fichier unique `combined.cer`

Exécutez la commande suivante avec tous les certificats CA que vous avez extraits précédemment :

- en Windows:

  ```
  type intermediateCA.cer rootCA.cer > combined.cer
  ```

- en Linux:

  ```
  cat intermediateCA.cer rootCA.cer >> combined.cer
  ```

Le certificat qui en résulte doit ressembler à ce qui suit :

```
-----BEGIN CERTIFICATE-----
MIIGbzCCBFegAwIBAgIICZftEJ0fB/wwDQYJKoZIhvcNAQELBQAwfDELMAkGA1UE
BhMCVVMxDjAMBgNVBAgMBVRleGFzMRAwDgYDVQQHDAdIb3VzdG9uMRgwFgYDVQQK
[redacted]
bRbjaT7JD6MBidAWRCJWC1R/5etTZwWwWrRCrzvIHC7WO6rCzwu69a+17ofCKlWs
y702dmPTKEdEfwhgLx0LxJr/Aw==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIF3TCCA8WgAwIBAgIIeyyb0xaAMpkwDQYJKoZIhvcNAQELBQAwfDELMAkGA1UE
BhMCVVMxDjAMBgNVBAgMBVRleGFzMRAwDgYDVQQHDAdIb3VzdG9uMRgwFgYDVQQK
[redacted]
oYYitmUnDuy2n0Jg5GfCtdpBC8TTi2EbvPofkSvXRAdeuims2cXp71NIWuuA8ShY
Ic2wBlX7Jz9TkHCpBB5XJ7k=
-----END CERTIFICATE-----
```

Le fichier résultat peut être utilisé en tant que `SAFE/web/conf/cacert.crt`

7.19 Problème persistant

⇨ voir l'index des messages

⇨ voir la section 8.5 qui décrit le Call Desk

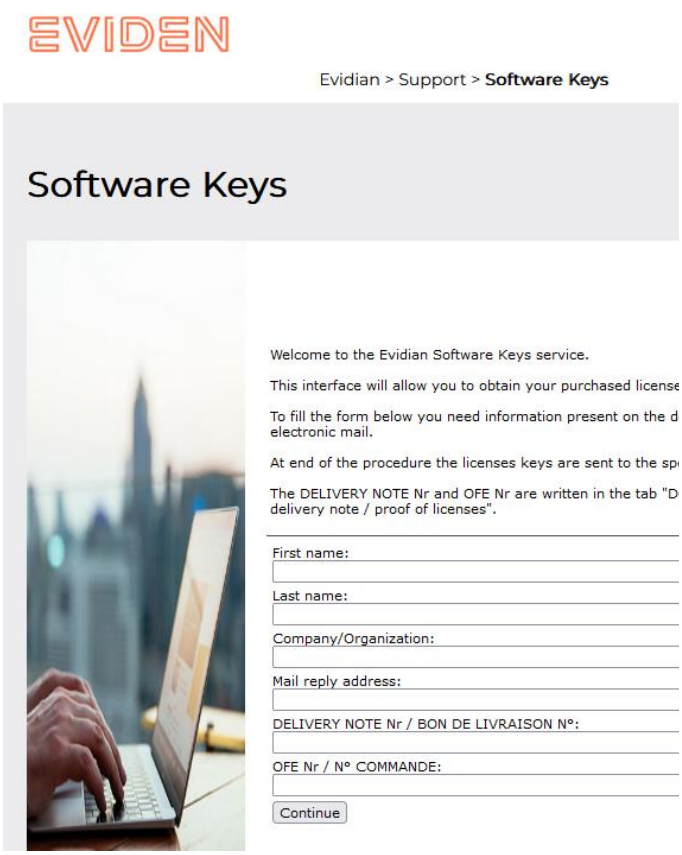
8. Accès au support Evidian

8.1 Page d'accueil du site support

- https://www.evidian.com/support
- Software Keys : obtenir des clés permanentes
- Registration : créer un compte
- Download : télécharger le produit ou uploader des snapshots
- Call desk : outil pour remonter un problème
- Knowledge Base : base de connaissance

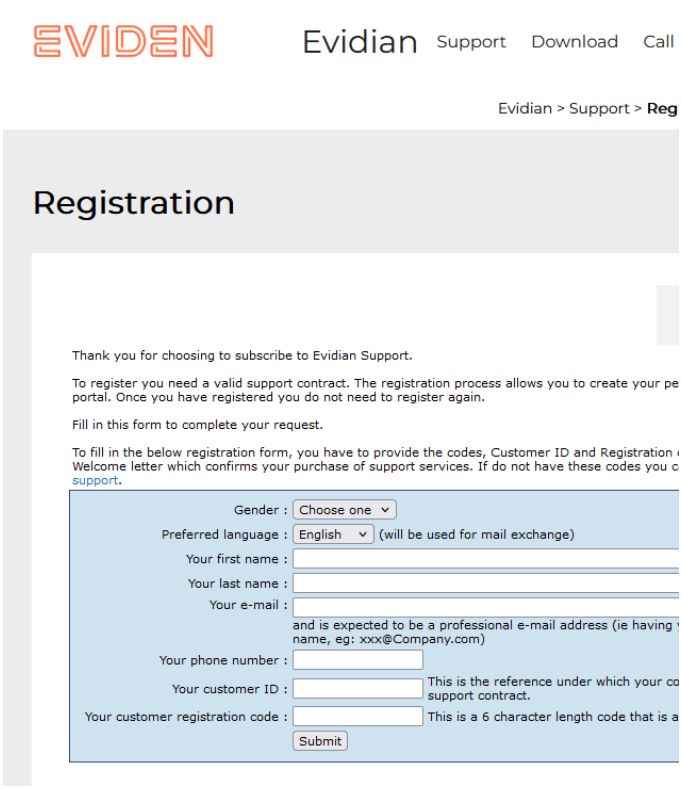
8.2 Clés de licence permanentes

- https://www.evidian.com/support/software-keys/
- Software Keys : obtenir des clés permanentes
- Remplir le formulaire à partir du bon de livraison envoyé pour donner suite à une commande
- Se munir des "hostname" et de l'OS des serveurs
- Pour obtenir une clé temporaire pour n'importe quel "hostname" et n'importe quel OS, voir la section 2.1.5



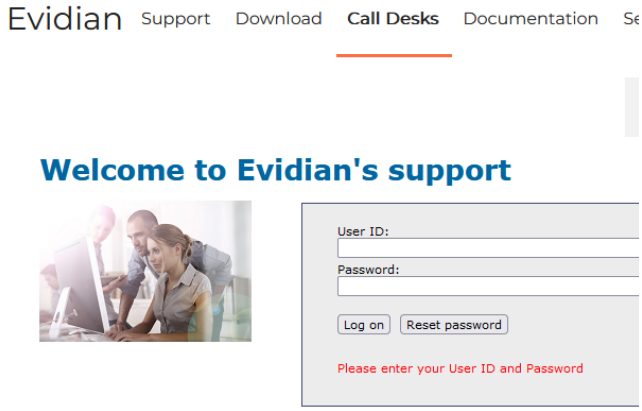
8.3 Créer un compte

- https://www.evidian.com/support/registration/
- Registration : créer un compte
- La procédure doit être exécutée une seule fois avec :
  - Votre identifiant client
  - Votre identifiant confidentiel
  - Une adresse e-mail unique
- Note : vos identifiants vous sont envoyés par mail si vous avez un contrat de support avec Evidian
- Ce que vous obtiendrez : un compte utilisateur et un mot de passe personnel sur le site



8.4 Accéder à votre compte

- https://www.evidian.com/support/call-desk/
- Se logger en haut à droite avec votre identifiant et mot de passe
- Vous avez alors accès à tous les services du site support



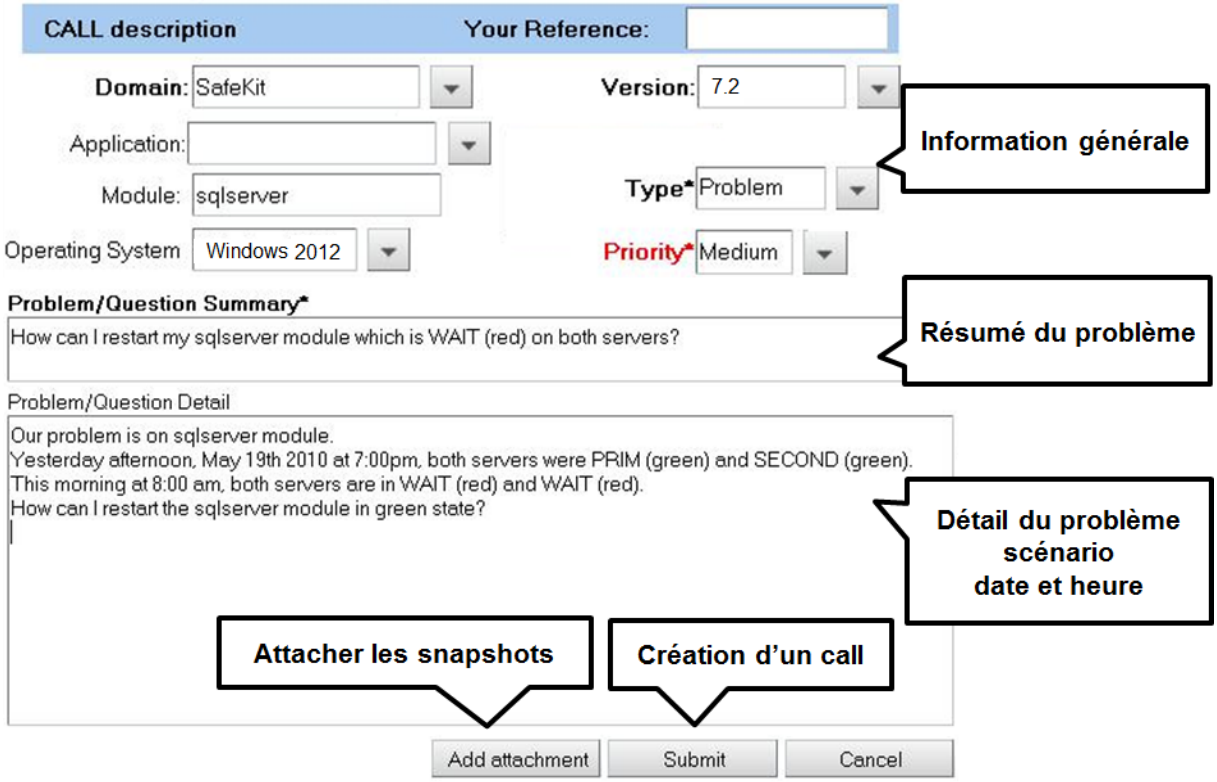
8.5 Le Call Desk pour remonter des problèmes

8.5.1 Les opérations du Call Desk

- https://www.evidian.com/support/call-desk/

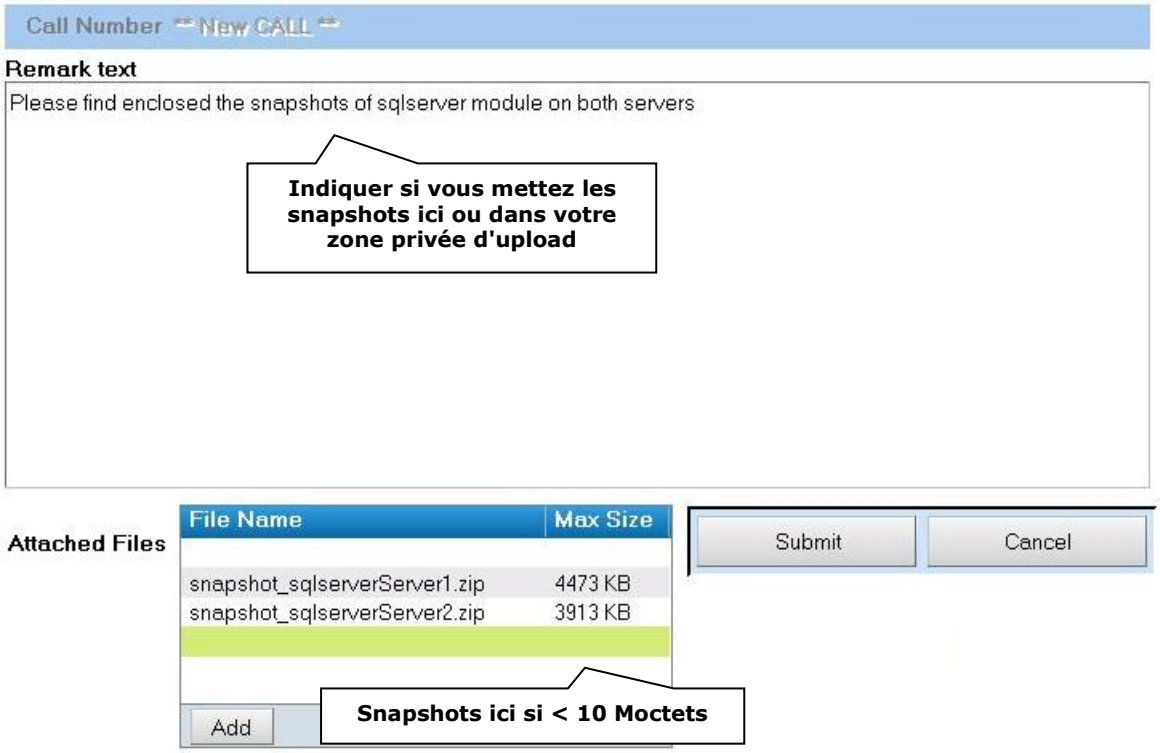
  Call desk : outil pour remonter un problème au support avec 2 opérations principales
- Création d'un Call
- Recherche d'un Call et échange avec le support sur un Call

8.5.2 Création d'un Call



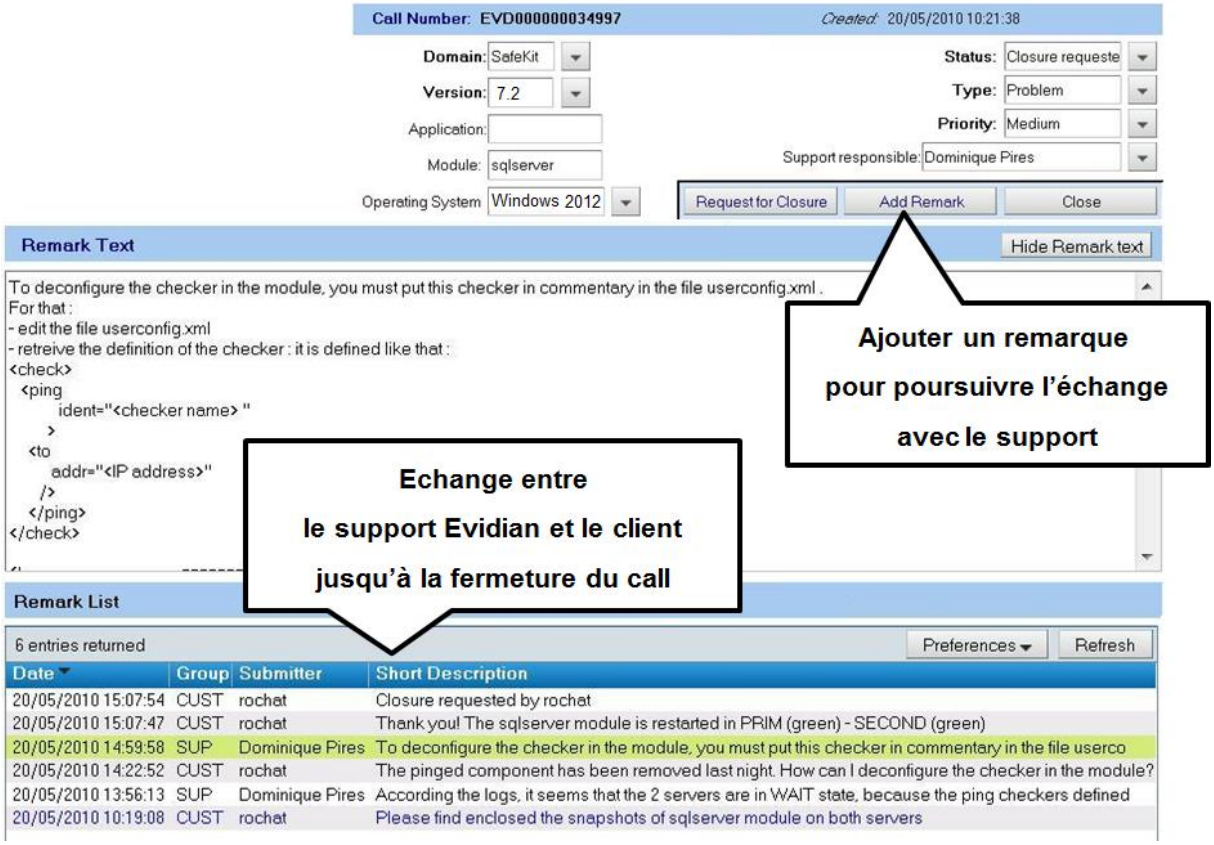
- Dans l'entête, préciser la version de Safekit, le type de problème et sa priorité ainsi que le nom du module et le l'OS de vos serveurs
- Résumer le problème puis le décrire plus en détail en précisant le scénario et la date et l'heure du problème
- Les snapshots du module qui pose un problème sont nécessaires pour l'analyse. Voir la page suivante pour attacher les snapshots
- Créer le call en appuyant sur "Submit"

8.5.3 Attacher les snapshots



- Lorsqu'un module SafeKit pose un problème, les snapshots du module sur tous les serveurs sont nécessaires pour l'analyse
- Pour récupérer les snapshots, voir la section 3.5
- Si la taille des snapshots est inférieure à 10 Moctets, vous pouvez les joindre en même temps que l'ouverture du call en cliquant sur "Add"
- Sinon, le temps de téléchargement des snapshots sur le site support peut durer plusieurs minutes. Dans ce cas indiquer dans "Remark text" que vous les téléchargez dans votre zone privée d'upload : voir section 8.6.3

8.5.4 Consultation des réponses au Call et échange avec le support

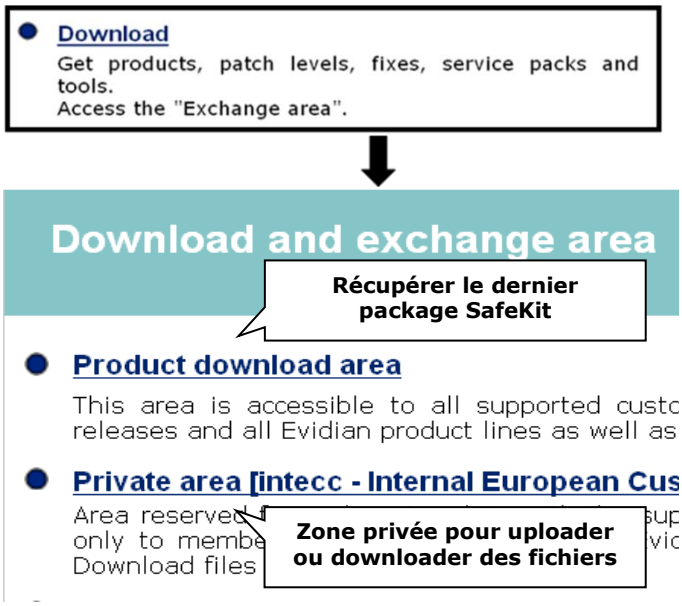


- Tous les échanges entre le support et le client se font au moyen de "Remarques"
- Quand le support ajoute une remarque sur un call, le client est averti par mail. C'est notamment le cas pour la première réponse du support après l'ouverture du call
- Après consultation de la dernière remarque du support, le client peut ajouter à son tour une nouvelle remarque
- L'échange a lieu jusqu'à la fermeture du call en accord entre le client et le support Evidian

8.6 Zone de download et d'upload de fichiers

8.6.1 2 zones de download et d'upload

- https://www.evidian.com/support/download/
- Product download area : zone de téléchargement des packages SafeKit
- Private area [identité du client] : zone privée d'upload de fichiers



8.6.2 La zone de download des packages produit

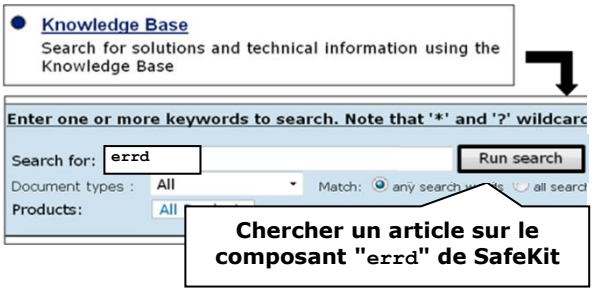
- Aller dans <Version 8.2>/Platforms/<Your platform>/Current versions
- Télécharger le package 64-bits SafeKit
- Pour plus d'information sur l'installation, la documentation et l'upgrade, voir la section 2

8.6.3 La zone privée d'upload

- Créer un répertoire pour un problème
- Uploader les snapshots dans ce répertoire avec
- Voir la section 3.5 pour la prise de snapshot
- Voir aussi la section 8.5.3 pour attacher un snapshot

8.7 Base de connaissances

- https://support.evidian.com/knowledge_base/
- Knowledge Base : base de connaissances
- Recherche par exemple de tous les articles sur le composant `errd` de SafeKit



9. Interface ligne de commande

L'interface ligne de commande SafeKit est fournie par la commande `safekit`. Pour l'utiliser :

En Windows	1. Ouvrir une console PowerShell en tant qu'administrateur 2. Aller à la racine du répertoire d'installation de SafeKit `SAFE` (par défaut `SAFE=C:\safekit` si `%SYSTEMDRIVE%`=C:) `cd c:\safekit` 3. Exécuter `.\safekit.exe <arguments>` pour la commande locale 4. Exécuter `.\safekit.exe -H "<hosts>" <arguments>` pour la commande distribuée sur plusieurs nœuds
En Linux	1. Ouvrir une console Shell en tant que root 2. Aller à la racine du répertoire d'installation de SafeKit `SAFE` (par défaut `SAFE=/opt/safekit`) `cd /opt/safekit` 3. Exécuter `./safekit <arguments>` pour la commande locale 4. Exécuter `./safekit -H "<hosts>" <arguments>` pour la commande distribuée sur plusieurs nœuds

9.1 Commandes de contrôle des services SafeKit

Utilisez les commandes suivantes pour démarrer/arrêter les services SafeKit et configurer leur démarrage automatique au boot.

9.1.1 Service `safeadmin`

Service principal de SafeKit obligatoire et démarré automatiquement au boot.

En Windows `net start safeadmin` `net stop safeadmin`	`safeadmin` peut aussi être contrôlé dans l'interface Services de Windows. Pour vérifier le status du service, exécuter :

	• dans une console command prompt `sc query safeadmin` • dans une console PowerShell: `Get-Service -name safeadmin`
En Linux `systemctl start safeadmin` `systemctl stop safeadmin`	Pour vérifier le status du service, exécuter : `systemctl status safeadmin`

9.1.2 Service safewebserver

Ce service nécessaire pour la console web SafeKit, le module checker et aux commandes distribuées. Par défaut, il est démarré automatiquement au boot. Pour plus de détails, voir la section 10.7.

`safekit webserver start` `safekit webserver restart` `safekit webserver stop`	Contrôle du service via la commande `safekit`.
En Windows `net start safewebserver` `net stop safewebserver`	Contrôle du service via la commande `net`. Pour vérifier le status du service, exécuter : • dans une console command prompt `sc query safewebserver` • dans une console PowerShell `Get-Service -name safewebserver`
En Linux `systemctl start safewebserver` `systemctl restart safewebserver` `systemctl stop safewebserver`	Contrôle du service via la commande `systemctl`. Pour vérifier le status du service, exécuter : `systemctl status safewebserver`
`safekit boot [webon | weboff | webstatus]`	Contrôle le démarrage automatique au boot du service `safewebserver` ("on" ou "off"). Par défaut : "on"

9.1.3 SNMP service

Service `Net-SNMP Agent` en Windows

La surveillance SNMP pour SafeKit n'est pas activée par défaut. Pour l'activer et le configurer voir la section 10.9.

En Windows, la surveillance SNMP est fournie par le service `Net-SNMP Agent`.

`safekit safeagent [start | stop | restart | check]`	Contrôle du service via la commande `safekit`.
`net start "Net-SNMP Agent"` `net stop "Net-SNMP Agent"`	Pour vérifier le status du service, exécuter : • dans une console command prompt `sc query "Net-SNMP Agent"` • dans une console PowerShell `Get-Service -name "Net-SNMP Agent"`
`safekit boot [snmpon | snmpoff | snmpstatus]`	Contrôle le démarrage automatique au boot du service `Net-SNMP Agent` ("on" ou "off"). Par défaut : "off"

Service standard `snmpd` en Linux

La surveillance SNMP pour SafeKit n'est pas activée par défaut. Pour l'activer et le configurer voir la section 10.9.

En Linux, la surveillance SNMP est fournie par l'agent `snmp` standard du système.

`systemctl start snmpd` `systemctl stop snmpd`	Pour vérifier le status du service, exécuter : `systemctl status snmpd`

9.2 Commandes de configuration et surveillance du cluster

`safekit cluster config [filepath de .xml ou .zip][lock|unlock]`	Applique la nouvelle configuration du cluster SafeKit avec le contenu du fichier passé en argument, `cluster.xml` ou `cluster.zip` : • `cluster.xml` Configure avec le fichier xml passé en argument et génère de nouvelles clés • `cluster.zip` Configure avec le `cluster.xml` et les clés contenues dans le .zip Appelée sans argument, cette commande conserve la configuration courante mais génère de nouvelles clés. Si cette commande est appelée avec le paramètre `lock`, les futurs appels ne seront autorisés que si le paramètre `unlock` est utilisé. Exemple : `safekit cluster config /tmp/newcluster.xml`

	 A utiliser avec prudence : le nouveau fichier `cluster.xml` et les clés de chiffrement doivent être impérativement recopiés sur les autres nœuds, de façon à s'assurer que tous les nœuds ont bien la même configuration de cluster et les mêmes clés.
`safekit cluster confcheck filepath`	Contrôle la configuration du cluster, avec le contenu du fichier xml passé en argument, sans l'appliquer
`safekit cluster confinfo`	Retourne, pour chaque serveur actif du cluster : • la date de dernière configuration du cluster, • la signature digitale de la dernière configuration du cluster. • L'état de verrouillage (1 pour `lock`, 0 pour `unlock`) de la commande de configuration. Cette commande permet de vérifier que tous les nœuds d'un cluster ont bien la même configuration de cluster. Exemple : Nœud Signature Date Verrou rh6server7 6f1032b11a7b2 … 33e67c 2016-05-20T17:06:45 0 rh7server7 6f1032b11a4e0 … 33e67c 2016-05-20T17:06:45 0  Les configurations du cluster SafeKit doivent impérativement être les mêmes sur tous les nœuds appartenant au même cluster. Les configurations asymétriques ne sont pas supportées.
`safekit cluster deconfig`	Supprime la configuration du cluster ainsi que les clés associées.
`safekit cluster state`	Retourne l'état global du cluster Pour chaque module installé et pour chaque nœud actif du cluster cette commande liste : • nom du nœud, • nom du module, • mode du module (farm ou mirror), • nº interne d'id du module, • date de la dernière configuration, • signature digitale de la dernière configuration Cette commande liste quels modules sont installés et sur quels serveurs. La signature et date de dernière configuration permet de vérifier qu'un module a bien la même configuration sur tous les nœuds et, si ce n'est pas le cas, où est la configuration la plus récente.

`safekit cluster genkey`	Régénère les clés de chiffrement pour la communication globale Safekit. La configuration du cluster doit être réappliquée (avec `safekit -G`) pour que cette modification soit prise en compte.
`safekit cluster delkey`	Supprime les clés de chiffrement pour la communication globale. La configuration du cluster doit être réappliquée (avec `safekit -G`) pour que cette modification soit prise en compte.
`safekit -H "*" -G`	Distribue la configuration locale du cluster et les clés de chiffrement associées lorsqu'elles existent, sur tous les nœuds. Relance une résolution de nom DNS pour tous les noms spécifiés dans `cluster.xml` et le `userconfig.xml` des modules, sans arrêter les modules (quand cela est possible). Voir la section 9.7 pour des détails sur cette commande distribuée.

9.3 Commandes de contrôle des modules

Les commandes s'appliquent au module nommé *AM*, passé en argument avec l'option `-m`.

Lorsque la variable d'environnement `SAFEMODULE` est définie avec le nom du module, l'argument `-m` n'est pas nécessaire. Celle-ci est définie lors de l'exécution des scripts du module (voir section 14.2).

`safekit start -m` *AM*	Démarre le module
`safekit waitstart -m` *AM*	Attend la fin du démarrage du module
`safekit stop -m` *AM*	Arrête le module
`safekit shutdown`	Stoppe tous les modules en cours d'exécution et attend leur arrêt complet
`safekit waitstop -m` *AM*	Attend la fin de l'arrêt du module
`safekit waitstate -m` *AM* `STOP | ALONE | UP | PRIM | SECOND`	Attend que le module atteigne l'état stable demandé (`NotReady` ou `Ready`)
`safekit restart -m` *AM*	Applique les scripts d'arrêt puis de démarrage de l'application sans provoquer de basculement.

	• Pour un module miroir Équivalent à `stop_prim ; start_prim` • Pour un module ferme Équivalent à `stop_both ; start_both`
`safekit stopstart -m` *AM*	Contrairement au `restart`, le `stopstart` provoque l'arrêt complet du module et son redémarrage automatique. Si le module était `PRIM`, il y a basculement du module `PRIM` sur l'autre serveur. Équivalent à `safekit stop -m` *AM* `; safekit start -m` *AM* mais sans arrêt des checkers dont l'action est `wait`
`safekit forcestop -m` *AM*	Force l'arrêt du module lorsque le `stop` n'a pas d'effet
`safekit swap [nosync] -m` *AM*	Module miroir uniquement Échange les rôles des serveurs primaire et secondaire. Utiliser l'option `nosync` pour permuter les rôles sans synchronisation des répertoires répliqués
`safekit second [fullsync] -m` *AM*	Module miroir uniquement Force le module à démarrer en secondaire ; échec si l'autre serveur n'est pas primaire Utiliser l'option `fullsync` pour forcer une réintégration complète de tous les répertoires répliqués
`safekit prim -m` *AM*	Module miroir uniquement Force le module à démarrer en primaire ; échec si l'autre serveur est déjà primaire Voir le bon usage de cette commande en section 5.3
`safekit errd off -m` *AM* `safekit errd on -m` *AM*	Suspend/redémarre la détection d'erreur sur les processus du module définis dans la section <`errd`> du fichier `userconfig.xml` Utile si on veut arrêter l'application sans provoquer de fausse détection et reprise. • La ressource *usersetting.`errd`* reflète l'état courant. • Si le module est démarré, un effet de bord de cette commande est d'effectuer un `update` du module pour prendre en compte ce changement Avec SafeKit < 8.2, utiliser `safekit errd suspend|resume -m` *AM*

`safekit checker off -m AM` `safekit checker on -m AM`	Arrête ou démarre l'ensemble des checkers (interface, TCP, IP, custom…). Cette commande est utile lors d'opérations de maintenance ; lorsqu'il est connu que les checkers vont détecter une panne à la suite d'un arrêt d'une partie de l'infrastructure informatique et qu'un basculement de serveur SafeKit n'est pas souhaitée. Note : • Ne peut être utilisée que sur un module démarré et dans un état stable (`ALONE`, `UP`, `PRIM`, `SECOND`, `WAIT`). • La ressource *usersetting.checker* reflète l'état courant. • Si le module est démarré, un effet de bord de cette commande est d'effectuer un `update` du module pour prendre en compte ce changement
`safekit failover off -m AM` `safekit failover on -m AM`	Permet de reconfigurer dynamiquement l'attribut *failover* du tag service du fichier de configuration (voir section 13.2.3). Note : • Ne peut être utilisée que sur un module miroir démarré et dans un état stable (`ALONE`, `PRIM`, `SECOND`, `WAIT`). • La ressource *usersetting.failover* reflète l'état courant. • Cette commande doit être exécutée sur tous les nœuds du module. Si ce n'est pas le cas le comportement n'est pas spécifié. • Si le module est démarré, un effet de bord de cette commande est d'effectuer un `update` du module pour prendre en compte ce changement
`safekit set -m AM -r <resource> -v <state>` `[-n] [-l]` `[-i identity]`	Utiliser cette commande pour affecter la valeur d'une ressource dans un custom checker. Par exemple : `safekit set -r custom.myresource -v up -m AM` `safekit set -r custom.myresource -v down -m AM` Chaque affectation des principales ressources est mémorisée dans un journal afin de conserver l'historique leur état. Utiliser `-n` pour désactiver la journalisation de l'affectation en cours ou `-l` pour la forcer. Utiliser `-i` pour spécifier l'identité du composant, qui affecte la ressource, dans le message logué.

Les commandes `restart`, `stop`, `stopstart` et `swap` acceptent également l'argument `-i identity`. Cet argument est défini lorsque l'action est appelée par les checkers ou la failover machine à des fins de journalisation et pour incrémenter le compteur `maxloop`. Quand il n'est pas présent, le compteur `maxloop` est réinitialisé.

9.4 Commandes de surveillance des modules

Les commandes s'appliquent au module nommé *AM*, passé en argument avec l'option `-m`.

Lorsque la variable d'environnement `SAFEMODULE` est définie avec le nom du module, l'argument `-m` n'est pas nécessaire. Celle-ci est définie lors de l'exécution des scripts du module (voir section 14.2).

`safekit level [-m AM]`	Indique la version de SafeKit et la licence Avec le paramètre *AM*, le script `level` du module est exécuté et ses résultats affichés
`safekit state`	Affiche l'état de tous les modules
`safekit state -m AM [-v | -lq]`	Affiche l'état du module *AM* Avec l'option verbose `-v`, les états de toutes les ressources du module sont listés : voir l'utilité des ressources dans la section 7.9 Avec l'option `-lq`, la commande retourne l'état (et le code d'exit): `STOP (0), WAIT (1), ALONE (2), UP (2), PRIM (3), SECOND (4)`
`safekit log -m AM [-s nb] [-A] [-l en|fr]`	Affiche les <nb> derniers messages E(vènement) du journal du module *AM*. Utiliser l'option `-A` pour afficher tous les messages (y compris les messages de debug). Sélectionner la langue avec l'option `-l`, `en` (Anglais) ou `fr` (Français). Par défaut : `-s 300`
`safekit logview -m AM [-A] [-l en|fr]`	Visualise en temps réel les derniers messages E(vènement) du journal du module *AM*. Utiliser `-A` pour afficher le journal verbeux (tous les messages y compris les messages de debug). Sélectionner la langue avec l'option `-l`, `en` (Anglais) ou `fr` (Français).
`safekit logview -m AM [-A] [-l en|fr]-s 300`	Visualise à partir des 300 derniers messages
`safekit logsave -m AM [-A] [-l en|fr] /tmp/f.txt`	Sauvegarde les messages E(vènement) du journal du module *AM* dans /tmp/f.txt (chemin absolu obligatoire). Utiliser `-A` pour sauvegarder tous les messages (y compris les messages de debug). Sélectionner la langue avec l'option `-l`, `en` (Anglais) ou `fr` (Français).

`safekit printi|printe -m AM "message"`	Les scripts applicatifs start/stop peuvent écrire des messages dans le journal du module avec un niveau I ou E.

9.5 Commandes de configuration des modules

Les commandes s'appliquent au module nommé *AM*, passé en argument avec l'option `-m`.

Lorsque la variable d'environnement `SAFEMODULE` est définie avec le nom du module, l'argument `-m` n'est pas nécessaire. Celle-ci est définie lors de l'exécution des scripts du module (voir section 14.2).

`safekit config -m AM`	A exécuter après avoir modifié un fichier sous `SAFE/modules/AM` tel que `userconfig.xml`, `start_prim/both` ou `stop_prim/both` (miroir/ferme). Il est recommandé d'appliquer cette commande lorsque le module est arrêté. Cependant, elle est permise dans les états `ALONE (Ready)`, ou `STOP` et `WAIT (NotReady)`. Dans ce cas, il s'agit d'une configuration dynamique où seul un sous-ensemble de paramètres peut être modifié. Les paramètres qui sont modifiables dynamiquement sont indiqués dans la section 13.
`safekit module genkey -m AM`	Génère les clés de chiffrement associées au module *AM*. Pris en compte à la prochaine configuration du module.
`safekit module delkey -m AM`	Efface les clés de chiffrement associées au module *AM*. Pris en compte à la prochaine configuration du module.
`safekit "*" -E AM`	Distribue la configuration locale du module *AM* et les clés de chiffrement associées lorsqu'elles existent, sur tous les nœuds du cluster. Voir la section 9.7 pour des détails sur cette commande distribuée.
`safekit deconfig -m AM`	A exécuter avant désinstallation du module Demande à chaque plugin défini dans `userconfig.xml` <errd>, <vip>, <rfs>, <user>... de prendre en compte la déconfiguration du module
`safekit confinfo -m AM`	Affiche des informations sur la configuration active et la configuration courante du module *AM* : • la configuration active est la dernière configuration appliquée avec succès. Elle est sous `SAFE/private/modules/AM` • la configuration courante est celle présente sous `SAFE/modules/AM`. Elle est différente de l'active lorsqu'elle a été modifiée sans avoir encore été appliquée.

	Cette commande est utile pour contrôler la configuration du module. Elle affiche : • la signature et la date de dernière modification (timestamp Unix) de la configuration active • la signature et la date de dernière modification (timestamp Unix) de la configuration courante Si les signatures sont différentes, cela signifie que les configurations ne sont pas identiques et qu'il est probablement nécessaire d'appliquer la configuration courante. Vous pouvez exécuter cette commande sur tous les nœuds qui implémentent le module, pour contrôler qu'ils ont bien la même configuration.
`safekit confcheck -m` *AM*	Contrôle la configuration du module sous `SAFE/modules/`*AM* sans l'appliquer
`safekit module install -m` *AM* `[-r] [-M id] [`*AM.safe*`]`	Installe le module *AM.safe* avec le nom *AM* `[-r]` force la réinstallation du module `[-M id]` force l'installation du module avec l'`id` spécifié comme module id • Le fichier *AM.safe* est cherché dans le répertoire `SAFE/Application_Modules` et ses sous répertoires. • Si le nom de fichier *AM.safe* n'est pas renseigné, le fichier *AM.safe* est cherché dans `SAFE/Application_Modules` • Un chemin absolu peut aussi être donné.
`safekit module package -m` *AM* `/…/newAM.safe`	Package le module *AM* dans `/…/newAM.safe` (chemin absolu obligatoire) Commande utilisée par la console pour créer un backup dans `SAFE/Application_Modules/backup/`
`safekit module uninstall -m` *AM*	Désinstalle le module *AM*. Détruit le répertoire de configuration du module `SAFE/modules/`*AM*
`safekit module list`	Liste les noms des modules installés
`safekit module listid`	Liste les noms et les ids des modules installés
`safekit module getports -m` *AM* `(or -i id)`	Liste les ports de communication qui synchronisent le module entre les serveurs
`safekit boot -m` *AM* `[on | off | status]`	Démarre automatiquement au boot ou non le module *AM* ("on" ou "off" ; par défaut "off"). En Windows, vous aurez peut-être également besoin d'appliquer la procédure décrite en section 10.4. Sans l'option `-m` *AM*, `safekit boot status` liste l'état de tous les modules au boot.

	Le démarrage au boot d'un module peut être défini dans la configuration du module avec l'attribut `boot` du tag `service` dans `userconfig.xml`. Cette option de configuration rend obsolète la commande `safekit boot -m AM on | off`. Toutefois, celle-ci est toujours supportée et remplace la configuration du module, à condition que l'attribut `boot` ne soit pas présent ou défini avec la valeur `ignore`.

9.6 Commandes de support

Les commandes s'appliquent au module nommé *AM*, passé en argument avec l'option `-m`.

Lorsque la variable d'environnement `SAFEMODULE` est définie avec le nom du module, l'argument `-m` n'est pas nécessaire. Celle-ci est définie lors de l'exécution des scripts du module (voir section 14.2).

`safekit snapshot -m AM /tmp/snapshot_xx.zip`	Sauvegarde le snapshot du module *AM* dans `/tmp/snapshot_xx.zip` (chemin absolu obligatoire) Un snapshot crée un dump puis récolte sous `SAFEVAR/snapshot/modules/AM` les 3 derniers dumps et les 3 dernières configurations du module pour les mettre dans le fichier .zip Pour analyser les snapshots, voir la section 7.16 Pour envoyer les snapshots au support, voir la section 8
`safekit dump -m AM`	Pour relever un problème en temps réel sur un serveur, générer un dump du module *AM* Un dump créé un directory `dump_year_month_day_hour_mn_sec` du côté serveur sous `SAFEVAR/snapshot/modules/AM`. Le directory `dump` contient les logs et l'état du module et ainsi que des informations sur l'état du système et des processus SafeKit au moment du dump
`safekit -r "commande spéciale"`	Exécute une commande sous `SAFEBIN` après avoir instancié les variables d'environnement SafeKit
`safekit clean [all | log | process | resource] [-m AM]`	Réinitialise les journaux, le fichier de ressources et arrête les principaux processus associés au module. Cette commande doit être utilisée avec précautions puisqu'elle détruit des fichiers de travail et arrête des processus du module. • `safekit clean log -m AM` Détruit les journaux du module (verbeux et non verbeux). A utiliser si les journaux sont corrompus (exemple : erreurs retournées lors de l'affichage du journal). • `safekit clean resource -m AM`

	Réinitialise le fichier de ressources du module. A utiliser si ce fichier est corrompu (exemple : erreurs retournées lors de l'affichage des ressources). • `safekit clean process -m AM` Arrête les principaux processus du module (heart). A utiliser lorsqu'à l'issue du stop et du forcestop du module des processus n'ont pas été arrêtés. • `safekit clean all -m AM` Valeur par défaut. « Nettoie » les journaux, le fichier de ressources et les processus.

9.7 Commandes distribuées sur plusieurs serveurs SafeKit

SafeKit offre une ligne de commande permettant son exécution sur plusieurs serveurs SafeKit. Chacun des serveurs doit exécuter le service web de SafeKit (voir section 10.7).

Le mot de passe affecté à l'initialisation du service web de SafeKit, doit être le même sur tous serveurs, même s'ils n'appartiennent pas au même cluster SafeKit.

La commande distribuée s'applique aux serveurs spécifiés avec l'argument `-H "<hosts>"` décrit ci-dessous.

`-H "*"` `-H "<cluster node names list>"`	La commande s'applique à des nœuds définis dans la configuration locale du cluster (voir section 12). Le protocole et le port sont ceux définis dans la configuration locale du service web de SafeKit. Par défaut, protocole `http` et port `9010`. • `-H "*"` Pour tous les nœuds du cluster • `-H "<cluster node names list>"` Liste des noms de nœuds tels qu'ils sont définis dans la configuration locale du cluster, avec la virgule comme séparateur. Par exemple : `-H "node1,node2"`
`-H "[[protocol:port],] <servers list>"`	La commande s'applique aux serveurs listés, qui n'appartiennent pas forcément au cluster local. • Spécification optionnelle du protocole (`http` ou `https`) et du port à utiliser. Par exemple : Si ce n'est pas spécifié, le protocole et le port sont ceux définis dans la configuration locale du service web de SafeKit. Par défaut, protocole `http` et port `9010`. • Liste des serveurs SafeKit (adresse IP ou nom) avec la virgule comme séparateur.

	Exemples: `-H "10.0.0.107"` `-H "[http:9500],10.0.0.107,10.0.0.108"` `-H "[https],S1.company.com,S2.company.com"`
`-H "<urls list>"`	La commande s'applique aux URLs listées, qui n'appartiennent pas forcément au cluster local. Exemple : `-H "http://192.168.0.2:9010,http://192.168.0.3:9010"`

Les commandes distribuées sont les suivantes.

`safekit -H "<hosts>" <safekit command arguments>`	Exécute la commande `safekit` sur les serveurs spécifiés par `-H`. A peu près toutes les commandes SafeKit peuvent être appliquées sur une liste de serveurs. Les exceptions sont les commandes `safekit logview`, `safekit -p` et `safekit -r` qui ne peuvent être exécutées que localement. Exemples : `safekit -H "http://192.168.0.2:9010,http://192.168.0.3:9010" level` `safekit -H "*" cluster confinfo` `safekit -H "node2" module list` `safekit -H "[http:9500],server1,server2" start -m AM`
`safekit -H "<hosts>" -E AM`	Exporte la configuration du module nommé *AM* sur les serveurs spécifiés par `-H`. Le module *AM* doit être installé localement. Cette commande réalise les actions suivantes : • Crée *AM*`.safe` à partir du module local `SAFE/modules/`*AM* et note son id • Transfère et installe *AM*`.safe` sur les serveurs distants • Installe le module sur les serveurs distants avec l'id local du module • Si le module était configuré localement, configure le module sur les serveurs distants Voir l'exemple d'utilisation dans la section 9.8.3.
`safekit -H "<hosts>" -G`	Exporte la configuration locale du cluster sur les serveurs spécifiés par `-H`. Cette commande réalise les actions suivantes :

	• Collecte le contenu du répertoire `SAFEVAR/cluster` • Transfère les fichiers collectés dans le répertoire `SAFEVAR/cluster` des serveurs distants • Déclenche le rechargement de la configuration de `safeadmin`. Voir l'exemple dans la section 12.2.2.

9.8 Exemples

9.8.1 Commande locale et distribuée

Par exemple, pour afficher la version (SafeKit, OS…) :

- Pour le nœud local

```
safekit level
```

- Pour tous les nœuds configurés dans le cluster SafeKit

```
safekit -H "*" level
```

9.8.2 Configuration globale du cluster

Voir la section 12.2.2.

9.8.3 Configuration globale d'un module

Les commandes en ligne équivalentes à l'assistant de configuration du module sont listées ci-dessous. Remplacer *AM* par le nom de votre module ; `node1` et `node2` par le nom de vos nœuds tels que définis lors de la configuration du cluster SafeKit.

1. Se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes

   Se connecter par exemple sur `node1`

2. aller dans le répertoire `SAFE`

   `SAFE` vaut `C:\safekit` en Windows quand `%SYSTEMDRIVE%`=C:, `/opt/safekit` en Linux

3. Optionnel

   A la première configuration uniquement, exécuter `safekit module install -m AM SAFE/Application_Modules/generic/mirror.safe`

   pour installer un nouveau module nommé *AM* à partir du modèle `mirror.safe`.

   Cela n'est pas nécessaire lors de la reconfiguration d'un module déjà installé.

4. Éditer la configuration du module et les scripts sous `SAFE/modules/AM/conf` et `SAFE/modules/AM/bin`

5. Optionnel

   Exécuter `safekit module genkey -m AM` ou `safekit module delkey -m AM`

   pour créer ou détruire les clés d'encryption du module.

   Il n'est pas nécessaire de créer une nouvelle clé d'encryption à la reconfiguration du module.

6. Exécuter `safekit -H "node1,node2" -E AM`

   pour (ré)installer le module *AM* et appliquer sa configuration qui est stockée sur le nœud qui exécute cette commande (`node1` dans cet exemple). Elle est appliquée sur tous les nœuds listés (`node1` et `node2`).

9.8.4 Snapshot d'un module

Ci-dessous, remplacer *AM* par le nom de votre module ; `node1` et `node2` par le nom de vos nœuds tels que définis lors de la configuration du cluster SafeKit.

1. se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes

   se connecter par exemple sur `node1`

2. aller dans le répertoire `SAFE`

   `SAFE` vaut `C:\safekit` en Windows quand `%SYSTEMDRIVE%`=C:, `/opt/safekit` en Linux

3. Exécuter `safekit snapshot -m AM /tmp/snapshot_nodes1_AM.zip`

   Pour sauvegarder le snapshot du module *AM* localement (cad sur *node1*) dans `/tmp/snapshot_xx.zip`.

Répéter ces commandes sur tous les nœuds du module

10.Administration avancée

10.1 Variables d'environnement et répertoires SafeKit

10.1.1 Global

Variable	Description
`SAFE` (rendu par `safekit -p`)	Répertoire d'installation de SafeKit : **`SAFE=/opt/safekit`** sur Linux et **`SAFE=C:\safekit`** sur Windows si SystemDrive=C: La licence est sous **`SAFE/conf/license.txt`**
`SAFEVAR` (rendu par `safekit -p`)	Répertoire des fichiers de travail de SafeKit : **`SAFEVAR=C:\safekit\var`** sur Windows et **`SAFEVAR=/var/safekit`** sur Linux
`SAFEBIN` (rendu par `safekit -p`)	Répertoire d'installation des binaires SafeKit : **`C:\safekit\private\bin`** sur Windows et **`/opt/safekit/private/bin`** sur Linux. Utile pour accéder aux commandes spéciales de SafeKit
`SAFE/Application_Modules`	Répertoire des modules .safe installables. Une fois un module installé, le module se trouve sous **`SAFE/modules`**

10.1.2 Module

Variable	Description
`SAFEMODULE`	Nom du module. La commande `safekit` n'a plus besoin du paramètre nom de module (`-m` *`AM`* = `-m SAFEMODULE`)

SAFE/Application_Modules	Répertoire des modules contenant les .safe installables. Une fois un module installé, le module se trouve sous **SAFE/modules**
SAFE/modules/*AM* et SAFEUSERBIN	L'édition d'un module, nommé *AM*, et de ses scripts se fait dans le répertoire **SAFE/modules/*AM***. On y trouve le fichier **userconfig.xml** du module et les scripts de démarrage et d'arrêt applicatif **start_prim**, **stop_prim** pour un miroir, **start_both**, **stop_both** pour une ferme (édition en direct ou via la console SafeKit) Après une configuration du module, les scripts sont copiés dans le répertoire d'exécution dans **SAFE/private/modules/*AM*/bin** : c'est la valeur de SAFEUSERBIN (ne pas modifier les scripts à cet endroit)
SAFEVAR/modules/*AM* et SAFEUSERVAR	Répertoire des fichiers de travail d'un module, nommé *AM* (SAFEUSERVAR=**SAFEVAR/modules/*AM***) Les messages d'output des scripts de démarrage et d'arrêt applicatif sont dans le fichier SAFEVAR/modules/*AM*/userlog_<year>_<month>_<day>T<time>_<script name>.ulog. Permet de vérifier s'il y a des erreurs au démarrage ou à l'arrêt de l'applicatif. Le userlog peut être désactivé dans le userconfig.xml du module avec : <user logging="none">.
SAFEVAR/snapshot/modules/*AM*	Répertoire des dumps et des configurations remontés dans un snapshot pour le module nommé *AM*. Voir section 9.6

L'arborescence des modules (empaqueté dans un .safe ou installé dans SAFE/modules/*AM*) est le suivant :

AM	Nom du module applicatif
conf	
userconfig.xml	Fichier XML de configuration utilisateur
userconfig.xml.template	Usage interne uniquement
modulekey.p12	

	Optionnel. Usage interne uniquement (chiffrement des communications internes du module)
modulekey.dat	Optionnel. Usage interne uniquement (chiffrement des communications internes du module)
bin	
prestart	Script du module exécuté au démarrage du module
start_prim or start_both	Script du module pour démarrer l'application en miroir ou ferme
stop_prim or stop_both	Script du module pour arrêter l'application en miroir ou ferme
poststop	Script du module exécuté à l'arrêt du module
web	
index.html	Obsolète (fichier pour la console web de SafeKit < 8)
manifest.xml	Usage interne uniquement

Depuis SafeKit 8, vous ne pouvez plus personnaliser l'affichage de la configuration rapide du module (puisque `index.html` est obsolète).

10.2 Services et démons SafeKit

Pour la liste détaillée des processus et ports de SafeKit, voir la section 10.3.3.1 et la section 10.3.3.2.

10.2.1 Services SafeKit

En Windows, les noms de processus ont comme extension `.exe`.

safeadmin (processus `safeadmin`)	Service principal de SafeKit obligatoire et démarré automatiquement au boot.
safewebserver (processus `httpd`)	Service pour la console web SafeKit, les module checkers, et la commande `safekit` globale (avec l'option `-H`)

Net-SNMP Agent (processus `safeagent`)	Agent SNMP SafeKit (optionnel, uniquement en Windows)

Pour les commandes de contrôle des services SafeKit, voir section 9.1.

10.2.2 Démons SafeKit par module

En Windows, les noms de processus ont comme extension `.exe`.

`heart`	Gère l'automate d'états du module et les procédures de reprise
`errd` `ipcheck` `intfcheck` `tcpcheck` `pingcheck` `modulecheck`	Checkers qui gèrent la détection d'erreurs
`vipd`	Synchronise une ferme de serveurs
`arpreroute`	Gère les requêtes arp pour l'IP virtuelle
`nfsadmin` `nfsbox` `reintegre`	Gèrent la réplication de fichiers en temps réel et la réintégration des données

10.3 Paramétrage du pare-feu

Si un pare-feu est actif sur le serveur SafeKit, il faut ajouter les règles autorisant les échanges réseau :

- Entre les serveurs pour les communications internes (échanges globaux et échanges spécifiques aux modules)
- Entre les serveurs et les stations de travail exécutant la console

Voici ci-dessous la commande pour configurer le pare-feu `Microsoft Windows` sur Windows ; `firewalld/iptables` sur Linux. Si vous avez choisi la configuration automatique du pare-feu pendant l'installation de SafeKit, cette commande déjà été exécutée.

`SAFE/private/bin/firewallcfg add` où `SAFE=C:\safekit` en Windows (si `%SYSTEMDRIVE%=C:`)	Sur tous les serveurs SafeKit : 1. Ouvrir une console PowerShell/shell en tant qu'administrateur/root 2. Exécuter `SAFE/private/bin/firewallcfg add` Cela configure le pare-feu du système.

`SAFE=/opt/safekit` en Linux	

Pour configurer d'autres pare-feu, référez-vous à la section 10.3.3 qui détaille les noms des processus SafeKit ainsi que les ports utilisés.

10.3.1 Paramétrage du pare-feu en Linux

Si la configuration automatique du pare-feu a été sélectionnée lors de l'installation de SafeKit, les commandes suivantes ne sont pas nécessaires.

Si la configuration automatique du pare-feu n'a été pas été faite, vous devez configurer le pare-feu.

Lorsque le pare-feu du système (firewalld/iptables) est utilisé, vous pouvez utiliser la commande `firewallcfg`. Celle-ci insère (ou supprime) les règles de pare-feu requises par les services SafeKit (`safeadmin` et `safewebserver`) et les modules.

Les administrateurs doivent s'assurer de l'absence de conflit avec une politique locale avant d'appliquer ces règles.

`SAFE/private/bin/firewallcfg add` `SAFE/private/bin/firewallcfg del` où `SAFE=/opt/safekit`	Ajout (ou suppression) des règles pour le pare-feu firewalld ou iptable pour les ports des services `safeadmin` et `safewebserver` • `SAFE/private/bin/firewallcfg add` Ajout des règles pour les services `safeadmin` et `safewebserver` • `SAFE/private/bin/firewallcfg del` Suppression des règles pour les services `safeadmin` et `safewebserver`
`SAFE/private/bin/firewallcfg add` *AM* `SAFE/private/bin/firewallcfg del` *AM* où `SAFE=/opt/safekit`	Ajout (ou suppression) des règles pour le pare-feu firewalld ou iptable pour les ports des modules SafeKit • `SAFE/private/bin/firewallcfg add` *AM* Ajout des règles pour le module nommé *AM* Cette commande doit être exécutée après l première configuration du module, puis au› configurations suivantes si celles-ci modifie ports utilisés (à vérifier avec la commande `safekit module getports -m` *AM*). • `SAFE/private/bin/firewallcfg del` *AM* Suppression des règles pour le module nommé *AM*

10.3.2 Paramétrage du pare-feu en Windows

Si la configuration automatique du pare-feu a été sélectionnée lors de l'installation de SafeKit, les commandes suivantes ne sont pas nécessaires.

Si la configuration automatique du pare-feu n'a été pas été faite, vous devez configurer le pare-feu.

Lorsque le pare-feu du système (`Microsoft`) est utilisé, vous pouvez utiliser la commande `firewallcfg`. Celle-ci insère (ou supprime) les règles de pare-feu requises par les services SafeKit (`safeadmin`, `safewebserver`, `safeacaserv et Net-SNMP Agent`) et les modules.

Les administrateurs doivent s'assurer de l'absence de conflit avec une politique locale avant d'appliquer ces règles.

`SAFE/private/bin/firewallcfg add` `SAFE/private/bin/firewallcfg del` où `SAFE=C:\safekit` (si `%SYSTEMDRIVE%=C:`)	Ajout (ou suppression) des règles pour le pare-feu de Microsoft • `SAFE/private/bin/firewallcfg add` Ajout des règles pour les services SafeKit et les modules • `SAFE/private/bin/firewallcfg del` Suppression des règles pour les services SafeKit et les modules

10.3.3 Autres pare-feux

Si vous utilisez un autre pare-feu ou souhaitez définir manuellement les règles de filtrage, cette partie liste les processus et ports utilisés par SafeKit afin d'aider à écrire les règles de pare-feu.

10.3.3.1 Liste des processus

10.3.3.1.1 Processus effectuant des communications internes

Les processus d'un module miroir

- `heart` : gère les procédures de récupération
- `errd` : détection d'absence de processus
- `nfsadmin`, `nfscheck` : gèrent la réplication de fichier

Les processus d'un module ferme

- `heart` : gère les procédures de récupération
- `errd` : détection d'absence de processus

10.3.3.1.2 Processus effectuant des communications externes

Les processus communs à tous les serveurs SafeKit, un processus par serveur et démarrés au boot :

- service `safeadmin` (processus `safeadmin`)

  processus central d'administration SafeKit. Obligatoire
- service `safewebserver` (processus `httpd`)

  service web pour la console, les module checkers et commande `safekit` globale (avec l'option `-H`)

- service `safecaserv` (processus `httpd`)

  service web pour sécuriser la console web avec la PKI de SafeKit (optionnel)
- En Windows : service `Net-SNMP Agent` (processus `safeagent`)

  agent SNMP v2 pour SafeKit (optionnel)

Les processus d'un module miroir

- `heart` : gère l'automate d'état du module
- `arpreroute` : gère les requêtes arp (envoie des paquets ARP)
- `nfsbox`, `reintegre` : gèrent la réplication de fichier
- `splitbraincheck` : gère la détection de split-brain (envoie des paquets ICMP ping)

Les processus d'un module ferme

- `vipd` : synchronise une ferme de serveurs
- `arpreroute` : gère les requêtes arp (envoie des paquets ARP)

Les processus pour un module miroir ou ferme selon la configuration des checkers

- `intfcheck` : test d'interface (configuration générée automatiquement lorsque <interface check=on>)
- `pingcheck` : ping d'une adresse (configuration <ping>)
- `ipcheck` : teste la présence d'une adresse IP locale (généré automatiquement lorsque la configuration <virtual_addr check=on> est présente)
- `modulecheck` : teste l'état d'un module SafeKit (configuration <module>)
- `tcpcheck` : teste l'établissement d'une connexion TCP (configuration <tcp>)

10.3.3.2 Liste des ports

Les ports suivants sont utilisés par SafeKit et les modules applicatifs.

10.3.3.2.1 Ports utilisés par les services

- `safeadmin`

  Par défaut, accès UDP distant sur le port 4800 (pour communiquer avec les `safeadmin` présents sur les autres serveurs SafeKit). Pour modifier la valeur du port, voir la section 12.1.3.
- `safewebserver`

  Accès TCP, local et distant, sur les ports 9010 par défaut pour la console web HTTP ou sur le port 9453 pour la console web HTTPS. Voir la section 10.7 pour la définition des valeurs des ports.

  Ce service est accédé localement, et à distance depuis les autres serveurs SafeKit et les stations de travail exécutant la console SafeKit.

- `safecaserv` (optionnel)

  Accès TCP, local et distant, sur le port 9001 par défaut. Pour la définition de la valeur du port, voir la section 11.3.1.9.4.

  Ce service est accédé localement, et à distance depuis les autres serveurs SafeKit et les stations de travail pour exécuter l'assistant de configuration HTTPS avec la PKI SafeKit.

- `Net-SNMP Agent` (Windows uniquement, optionnel)
  Accès UDP, local et distant, sur le port 3600 par défaut. Pour la définition de la valeur du port, voir la section 10.9.

10.3.3.2.2 Ports utilisés par les modules

Lorsqu'un module applicatif est configuré, on peut exécuter la commande `safekit module getports -m` *AM* pour lister les ports externes utilisés par le module *AM*. Le pare-feu doit être configuré pour ouvrir l'accès à ces ports. La valeur des ports est calculée automatiquement en fonction de l'id du module. La commande `safekit module listid` affiche le nom des modules installés et leur `id`.

Les règles suivantes permettent de calculer les valeurs des ports selon `id` du module. Lorsque des checkers sont configurés pour le module, il peut être nécessaire d'ajouter des règles selon la configuration des checkers. La communication locale (localhost) doit être autorisée pour tous les processus SafeKit.

Pour un module mirror

- Heart

  port UDP pour communiquer entre serveurs SafeKit

  port=8888 +(id-1)

- rfs (file replication)

  port TCP pour la réplication entre serveurs SafeKit

  safenfs_port=5600 +(id-1)x4

Exemple pour un module miroir avec l'id 1, avec la commande `safekit module getports -m mirror` :

```
List of the ports used by SafeKit

Process         Ports
safeadmin
      port    UDP 4800
webconsole
      port    TCP 9010
heart
      port    UDP 8888
rfs
     safenfs_port    TCP 5600
```

Pour un module farm

- Port utilisé par farm

port UDP pour communiquer entre serveurs SafeKit
port 4803 + (id-1)x3

Exemple pour un module farm avec l'id 2, , avec la commande `safekit module getports -m farm` :

```
List of the ports used by SafeKit

Process         Ports
safeadmin
        port    UDP 4800
webconsole
        port    TCP 9010
farm
        port    UDP 4806
```

Pour les checkers

- Ping checker
  Modifier les règles ICMP pour autoriser ping à destination de l'adresse définie dans la configuration.
- TCP checker
  Autoriser les connexions TCP connexions à destination de l'adresse définie dans la configuration <tcp>.
- Module checker
  Autoriser les connexions TCP à destination du port 9010 pour le serveur exécutant le module applicatif qui est testé.
- Splitbrain checker
  Modifier les règles ICMP pour autoriser ping à destination de l'adresse définie dans la configuration <splitbrain>.

10.4 Configuration au boot et au shutdown en Windows

Le service `safeadmin` est configuré pour démarrer automatiquement au boot et s'arrêter proprement au shutdown. A son tour, ce service démarre les modules configurés pour démarrer au boot et arrête les modules.

Sur certaines plateformes Windows, le démarrage au boot de `safeadmin` échoue car la configuration réseau n'est pas prête ; au shutdown, les modules n'ont pas le temps de s'arrêter proprement car le délai d'attente de l'arrêt du service est trop court. Si vous rencontrez ce type de problème, appliquez l'une des procédures suivantes.

Si vous utilisez l'agent SNMP de SafeKit, adaptez la procédure suivante pour positionner le démarrage manuel du service `Net-SNMP Agent` et inclure son démarrage/arrêt dans les scripts de démarrage (`safekitbootstart.cmd`) et arrêt de SafeKit (`safekitshutdown.cmd`).

10.4.1 Procédure automatique

1. Ouvrir une console PowerShell en tant qu'administrateur
2. cd `SAFE\private\bin\`
3. Exécuter le script `addStartupShutdown.cmd`

Ce script positionne le démarrage manuel de `safeadmin` et ajoute dans les objets stratégies de groupe, les scripts de démarrage (`safekitbootstart.cmd`) et d'arrêt (`safekitshutdown.cmd`) de SafeKit. Si le script échoue, appliquez la procédure manuelle.

10.4.2 Procédure manuelle

Vous devez appliquer la procédure suivante qui utilise l'éditeur d'objets de stratégie de groupe :

1. Positionner en démarrage manuel le service `safeadmin`
2. Ouvrir une console PowerShell en tant qu'administrateur
3. Lancer la console MMC à l'aide de la commande `mmc`
4. Fichier - Ajouter/Supprimer un composant logiciel enfichable ; Ajouter - "Editeur d'objets de stratégie de groupe" - OK
5. Sous "Racine de la console"/"Stratégie ordinateur local"/"Configuration ordinateur"/"Paramètres Windows"/"Scripts (démarrage/arrêt)", double cliquer sur "Démarrage". Cliquer sur ajouter puis entrer pour le nom du script : `c:\safekit\private\bin\safekitbootstart.cmd`. Ce script lance le service `safeadmin`.
6. Sous "Racine de la console"/"Stratégie ordinateur local"/"Configuration ordinateur"/"Paramètres Windows"/"Scripts (démarrage/arrêt)", double cliquer sur "Arrêt du système". Cliquer sur ajouter puis entrer pour le nom du script : `c:\safekit\private\bin\safekitshutdown.cmd`. Ce script arrête proprement tous les modules en cours d'exécution.

10.5 Paramétrage des antivirus

Les antivirus peuvent rencontrer des problèmes de détection avec SafeKit en raison de son intégration étroite avec le système d'exploitation, des mécanismes d'IP virtuelle, de la réplication en temps réel et du redémarrage des services critiques. Il peut alors être nécessaire de paramétrer l'antivirus pour exclure certains répertoires et processus. La liste des répertoires et processus est donnée ci-dessous.

Répertoires

`SAFE`	Répertoire d'installation de SafeKit : • En Windows `C:\safekit` si `SystemDrive=C:` • En Linux `/opt/safekit`
`SAFEVAR`	Répertoire des fichiers de travail de SafeKit : • En Windows `C:\safekit\var` si `SystemDrive=C:` • En Linux `/var/safekit`
Répertoires répliqués	Tous les répertoires répliqués définis dans les modules miroir

Processus

Les processus SafeKit pour les services et démons sont listés dans la section 10.2.

Les exécutables sont localisés sous :

`SAFE`	`safekit` command
`SAFE/private/plugin/*/*`	Exécutables qui sont lancés sur changement d'état d'un module
`SAFE/private/bin`	Exécutables SafeKit
`SAFE/web/bin`	Exécutables pour le SafeKit web service

10.6 Sécurisation des communications internes au module

Il est possible de sécuriser les communications internes au module entre les différents nœuds du cluster, en créant les clés de chiffrement associées au module. Par défaut, ces clés sont générées par SafeKit avec une autorité de certification « privée » (SafeKit PKI). Dans SafeKit <= 7.4.0.31, la clé générée a une durée de validité de 1 an. Voir la section 10.6.3.1 pour les solutions quand la clé expire.

Depuis SafeKit 7.4.0.16, vous pouvez également fournir vos propres clés générées avec votre autorité de certification de confiance (PKI d'entreprise ou PKI commerciale). Voir section 10.6.3.2 pour plus de détails.

Depuis SafeKit 7.4.0.32, le module peut être reconfiguré avec de nouvelles clés même dans l'état ALONE (reconfiguration dynamique).

Lorsque toutes les instances du module n'ont pas la même clé de chiffrement, la communication entre instances est impossible. Réappliquer la configuration contenant la clé valide sur tous les nœuds pour rétablir une configuration correcte.

Il est possible de visualiser la configuration en exécutant la commande `safekit confinfo -m AM` sur chaque nœud (voir section 9.5). Cette information est également affichée par la console web avant d'éditer la configuration du module et avant le démarrage global

La ressource **`encryption`** reflète le mode de communication courant du module : "on"/"off" lorsque le chiffrement est actif/inactif. Pour voir l'état des ressources, voir la section 7.3. Cette ressource se nomme `usersetting.encryption`.

10.6.1 Configuration avec la console web de SafeKit

Lors de la configuration du module avec la console Web SafeKit, le cryptage de la communication est activé à l'étape 3 de l'assistant de configuration du module (voir section 3.3.2).

10.6.2 Configuration en ligne de commandes

Les commandes équivalentes pour créer les clés de chiffrement associées à un module sont :

1. `safekit module genkey -m AM`
2. `safekit -H "server1,server2"  -E AM`

où server1 et server2 sont les nœuds qui implémentent le module

Les commandes équivalentes pour supprimer les clés de chiffrement associées à un module sont :

1. `safekit module delkey -m AM`
2. `safekit -H "server1,server2"  -E AM`

   où server1 et server2 sont les nœuds qui implémentent le module

Pour la description des commandes, voir la section 9.5.

10.6.3 Configuration avancée

SafeKit peut sécuriser la communication interne avec des certificats qui sont générés avec une autorité de certification « privée » (SafeKit PKI). Depuis SafeKit 7.4.0.16, vous pouvez également fournir vos propres certificats générés avec votre autorité de certification de confiance (PKI d'entreprise ou PKI commerciale).

10.6.3.1 Configuration avancée avec la PKI SafeKit

Dans SafeKit <= 7.4.0.31, la clé de chiffrement des communications a une durée de validité de 1 an. Quand celle-ci expire dans un module miroir avec la réplication de fichiers, la réintégration sur le secondaire échoue. Pour revenir à une situation normale, il faut reconfigurer le module avec une nouvelle clé comme décrit dans SK-0084. A partir de SafeKit > 7.4.0.31, la durée de validité est de 20 ans.

Si vous ne pouvez pas upgrader SafeKit, vous pouvez générer de nouvelles clés avec une période de validité plus longue. Pour cela, appliquez la procédure suivante :

1. Arrêter le module *AM* sur tous les nœuds
2. Sur l'un des nœuds, se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
3. Exécuter `safekit module genkey -m AM`
4. Supprimer le fichier `SAFE/modules/AM/conf/modulekey.p12`
5. Aller dans le répertoire SAFE`/web/bin`
6. Exécuter `./openssl req -config ../conf/ssl.conf -subj "/O=SafeKiModule/CN=mirror" -new -x509 -sha256 -nodes -days 3650 -newkey rsa:2048 -keyout pkey.key -out cert.crt`

   Affecter à l'argument `-days`, la nombre de jours que vous souhaitez comme durée de validité
7. Exécuter `./openssl pkcs12 -export -inkey ./pkey.key -in ./cert.crt -name "Module certificate" -out modulekey.p12`

   Cette commande nécessite de renseigner un mot de passe. Contactez le support Evidian pour obtenir la valeur correcte du mot de passe
8. Supprimer les fichiers `pkey.key` et `cert.crt`
9. Déplacer le fichier `modulekey.p12` sous `SAFE/modules/AM/conf`
10. Aller dans le répertoire `SAFE`
11. Exécuter `safekit -H "server1,server2" -E AM`

où server1 et server2 sont les nœuds qui implémentent le module

Le module est configuré sur les 2 nœuds avec sa nouvelle clé et prêt à être démarré.

10.6.3.2 Configuration avancée avec une PKI externe

Depuis SafeKit 7.4.0.16, vous pouvez fournir votre propre clé générée avec votre autorité de certification de confiance (PKI d'entreprise ou PKI commerciale).. Pour cela, appliquez la procédure suivante :

1. Arrêter le module *AM* sur tous les nœuds
2. Sur l'un des nœuds, se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
3. Exécuter `safekit module genkey -m AM`
4. Supprimer le fichier `SAFE/modules/AM/conf/modulekey.p12`
5. Ajoutez le certificat X509 au format PEM, pour votre autorité de certification (certificat de l'AC ou bundle de certificats de toutes les autorités de certification) au `fichier SAFE/web/conf/cacert.crt`
6. Aller dans le répertoire SAFE`/web/bin`
7. Générer votre certificat à l'aide de la PKI en spécifiant dans le sujet : `"/O=SafeKiModule/CN=mirror"`
8. Copier les fichiers générés `pkey.key` et `cert.crt` dans le répertoire `SAFE/web/bin`
9. Exécuter `./openssl pkcs12 -export -inkey ./pkey.key -in ./cert.crt -name "Module certificate" -out modulekey.p12`

   Cette commande nécessite de renseigner un mot de passe. Contactez le support Evidian pour obtenir la valeur correcte du mot de passe
10. Supprimer les fichiers `pkey.key` et `cert.crt`
11. Déplacer le fichier `modulekey.p12` sous `SAFE/modules/AM/conf`
12. Exécuter `SAFE/safekit -H "server1,server2" -E AM`

    où server1 et server2 sont les nœuds qui implémentent le module

Le module est configuré sur les 2 nœuds avec sa nouvelle clé et prêt à être démarré.

10.7 Configuration du service web de SafeKit

SafeKit livre le service web, `safewebserver`, qui s'exécute sur chaque serveur SafeKit. C'est un serveur Apache standard obligatoire pour :

- la console web (voir section 3)
- l'interface en ligne de commandes distribuées sur le cluster (voir section 9.7)
- les checkers de type <module> (voir section 13.16)

Le service `safewebserver` démarre automatiquement à la fin de l'installation du package SafeKit et au reboot des serveurs. Si vous n'avez pas besoin de ce service et souhaitez supprimer son démarrage automatique au boot, référez-vous à la section 9.1.

La configuration par défaut est HTTP avec authentification à base de fichiers, initialisée avec un seul utilisateur `admin` ayant le rôle Admin. Cela peut être changé via l'édition de fichiers de configuration.

10.7.1 Fichiers de configuration

La configuration de `safewebserver` est définie dans les fichiers livrés sous **`SAFE/web/conf`**. Il s'agit de fichiers de configuration Apache standards (voir `http://httpd.apache.org`). La configuration du service est décomposée dans plusieurs fichiers mais, pour les configurations les plus usuelles seul le fichier `httpd.conf` nécessite d'être modifié.

- Après modification, vous devez redémarrer le service pour charger la nouvelle configuration avec la commande : `safekit webserver restart` (voir section 9.1).
- Ne pas modifier les fichiers `.default` sous **`SAFE/web/conf`** car il s'agit de sauvegarde de la configuration livrée.

Le fichier `httpd.conf` est essentiellement constitué d'une sérié de `Define`. Le caractère de commentaire # désactive la définition.

Les principaux `Define` sont :

Définition du port de connexion :

```
Define httpport 9010
Define httpsport 9453
```

Définit les numéros de ports d'écoute en mode HTTP et HTTPS. Voir la section 10.7.2 pour leur utilisation.

Définition de l'authentification utilisateur

```
Define usefile
#Define useldap
#Define useopenid
…
```

Sélectionne l'authentification utilisateur voulue. Au plus, une seule doit être définie. Voir la section 11.4 pour plus de détails).

Définition de la journalisation Apache

Désactivé par défaut

```
#Define Loglevel info
#Define accesslog
```

Décommenter ces lignes pour activer la journalisation. Les journaux sont `httpd.log` et `access.log`. Ils sont générés dans le répertoire `SAFEVAR`.

Définition de la durée de validité d'une session :

```
Define SessionMaxAge 28800
```

Depuis SafeKit 8.2.1, l'utilisateur est automatiquement déconnecté après 8 heures d'inactivité (28800 secondes). Si nécessaire, ajustez cette valeur.

Les autres `Define` sont documenté dans le fichier `httpd.conf`.

Les autres fichiers de configuration sont listés ci-dessous. La modification de l'un d'entre eux peut causer des problèmes lors de la mise à jour de SafeKit.

Configuration globale	**httpd_main.conf**
Configuration de l'authentification à base de fichier	**httpd.webconsolefileauth.conf** Utilisation des fichiers `user.conf` et `group.conf` dans `SAFE/web/conf`
Configuration de l'authentification à base de formulaire	**httpd.webconsoleformauth.conf**
Configuration de l'authentification à base de serveur LDAP/AD	**httpd.webconsoleldap.conf** Utilisation d'un serveur LDAP/AA
Configuration de l'authentification à base de serveur OpenID Connect	**httpd.webconsoleopenidauth.conf** Utilisation d'un fournisseur d'identité OpenID connect
Configuration HTTPS	**httpd.webconsolessl.conf** (dans le sous-répertoire `ssl`) Utilisation du fichier `sslgroup.conf dans SAFE/web/conf`

10.7.2 Configuration des ports de connexion

Par défaut, connectez la console web avec l'URL `http://host:9010`. Le serveur web SafeKit redirigera vers la page appropriée en fonction de vos paramètres de sécurité.

Si vous devez modifier la valeur par défaut :

1. Éditez `SAFE/web/conf/httpd.conf` et modifiez la valeur des variables `httpport` ou `httpsport`
2. Redémarrez le service avec la commande `safekit webserver restart`

Les configurations HTTP et HTTPS ne doivent pas être activées simultanément. Voir la section 11.3 pour la configuration HTTPS.

La valeur par défaut `9010(HTTP)/9453(HTTPS)` est également utilisée le module checker. Par conséquent, dans la configuration des modules qui définissent un checker de type <module> :

1. Éditez le fichier `userconfig.xml` du module
2. Rajoutez l'attribut `port` et lui affecter la nouvelle valeur du port

```
<check>
    <module name="mirror">
```

```
     <to addr="192.168.1.31" port="9010"/>
    </module>
</check>
```

3. Appliquez la nouvelle configuration du module

10.7.3 Configuration de HTTP/HTTPS et de l'authentification utilisateur

- La configuration par défaut est pour HTTP. Elle inclut l'authentification à base de fichier, initialisée avec un seul utilisateur `admin` ayant le rôle Admin.
- La configuration HTTPS requiert l'installation de certificats ainsi qu'une méthode d'authentification des utilisateurs.

Pour une description détaillée et leur mise en œuvre, voir la section 11 .

Pour revenir à la configuration HTTP si celle-ci a été changée pour HTTPS, voir la section 11.2.1.1.

10.7.4 API SafeKit

Utilisez Swagger UI pour visualiser et interagir avec l'API SafeKit fournie par le service web de SafeKit. Pour cela, connectez un navigateur à l'URL http://host:9010/swagger-ui/index.html. Cela permet notamment de déboguer des problèmes avec la console web SafeKit et/ou l'API.

10.8 Notification par mail

Vous pouvez avoir besoin d'envoyer une notification, par exemple un courrier électronique, lorsque le module est démarré, arrêté ou qu'il exécute un basculement. Ceci est mis en œuvre grâce aux scripts du module.

Pour la notification par courrier électronique, vous devez d'abord choisir un programme en ligne de commande pour envoyer le courrier. Sous Windows, vous pouvez utiliser la commande `Send-MailMessage` de l'utilitaire Microsoft Powershell. Pour Linux, vous pouvez utiliser la commande `mail`.

- Notification du démarrage et de l'arrêt du module

  Les scripts de module `prestart`/`poststop` peuvent être utilisés pour envoyer une notification sur le démarrage/arrêt du module.

- Notification sur le basculement du module

  Le script de module transition peut être utilisé pour envoyer une notification sur les principaux changements d'état du module. Par exemple, il peut être utile de savoir quand le module miroir devient ALONE (lors d'un basculement par exemple).

Pour la description des scripts du module, voir la section 14.

Pour un exemple complet avec le module de démonstration `notification.safe`, voir la section 15.12.

10.9 Surveillance SNMP

SafeKit peut être surveillé via SNMP. Depuis la version 8, les implémentations pour Windows et Linux diffèrent. En Windows, SafeKit utilise son propre agent SNMP, alors qu'en Linux, l'agent SNMP du système est utilisé.

10.9.1 Surveillance SNMP en Windows

Pour utiliser l'agent SNMP de SafeKit, `Net-SNMP Agent`, vous devez :

1. le configurer pour démarrer au boot avec la commande :

`safekit boot [snmpon | snmpoff | snmpstatus]`	Contrôle le démarrage automatique au boot du service `Net-SNMP Agent` ("on" ou "off" ; par défaut "off")

2. ajouter la règle de pare-feu correspondante

   Si vous utilisez le pare-feu du système, le pare-feu a déjà été configuré si vous avez appliqué la commande :

   `SAFE/private/bin/firewallcfg add`

3. le démarrer avec la commande :

`safekit safeagent [start | stop | restart | check]`	Contrôle le démarrage/arrêt du service `Net-SNMP Agent` qui met en œuvre un agent SNMP SafeKit.

La configuration du service `Net-SNMP Agent` est définie dans le fichier auto-documenté **`SAFE/snmp/conf/snmpd.conf`**. C'est un fichier de configuration net-snmp standard décrit dans http://net-snmp.sourceforge.net. Par défaut, le service écoute sur le port UDP **`agentaddress`** 3600 et accepte des requêtes de lecture de la communauté publique et des requêtes d'écriture de la communauté privée. Les requêtes de lecture sont utilisées pour lire l'état d'un module alors que les requêtes d'écriture permettent de réaliser des actions sur le module.

Vous pouvez changer la configuration par défaut suivant vos besoins. Lorsque vous modifier **`snmpd.conf`**, vous devez redémarrer l'agent pour charger la nouvelle configuration : `safekit safeagent restart`.

Depuis SafeKit 8, le service se nomme `Net-SNMP Agent` au lieu de `safeagent` dans les versions antérieures.

10.9.2 Surveillance SNMP en Linux

Depuis la version 8.0, Safekit ne vient plus avec son propre agent snmp, aussi les commandes suivantes sont obsolètes en Linux : `safeagent install, safeagent start, safeagent stop, boot snmpon, boot snmpoff, boot snmpstatus`.

En remplacement, il est possible de configurer l'agent SNMP standard du système pour accéder la mib safekit :

1. Installer net-snmp
   `dnf install net-snmp net-snmp-utils`
2. Si selinux est en mode *enforced*, il doit être mis en mode *permissive* pour snmp :
   `semanage permissive -a snmpd_t`
3. Si le pare-feu est actif, le port snmp doit être ouvert :
   `firewall-cmd --permanent --add-service snmp`

   `firewall-cmd --reload`

4. Éditer `/etc/snmp/snmpd.conf`
   Ajouter les lignes suivantes :
   ```
   pass .1.3.6.1.4.1.107.175.10 /opt/safekit/snmp/bin/snmpsafekit
   view systemview included .1.3.6.1.4.1.107.175.10
   ```

   Note : La ligne "view systemview" assigne les droits d'accès. Il peut être nécessaire de la modifier suivant les contraintes locales.

5. Activer et démarrer l'agent snmp
   ```
   systemctl enable snmpd
   systemctl start snmpd
   ```

10.9.3 La MIB SafeKit

La MIB SafeKit est livrée dans `SAFE/snmp/mibs/safekit.mib` .

La MIB SafeKit est accessible avec l'identifiant suivant (OID, préfixe des variables SNMP de SafeKit SNMP): **= enterprises.bull.safe.safekit (1.3.6.1.4.1.107.175.10).**

La MIB SafeKit définit :

- la table de modules : `skModuleTable`

  L'index dans cette table correspond à l'id du module applicatif tel qu'il est retourné par la commande `safekit module listid`.

  A travers la MIB, vous pouvez lire et afficher l'état des modules applicatifs sur un serveur (`STOP, WAIT, ALONE, UP, PRIM, SECOND`) ou vous pouvez agir sur un module (`start`, `stop`, `restart`, `swap`, `stopstart`, `prim`, `second`).

  Par exemple, l'état du module d'id 1 est lu avec un get sur la variable SNMP suivante :
  ```
  enterprises.bull.safe.safekit.skModuleTable.skModuleEntry.skModuleCurrentState.1 = stop (0)
  ```

  Utiliser la commande `snmpwalk` pour voir l'ensemble des entrées de la MIB (commande non livrée avec le produit).

- La table de ressources : **`skResourceTable`**

  Chaque élément définit une ressource comme un checker d'interface réseau `"intf.192.168.0.0"` et son status (`unknown`, `init`, `up`, `down`).

  Exemple: requête SNMP get sur `enterprises.bull.safe.safekit.skResourceTable.skResourceEntry.skResourceName.1.2` veut dire nom de la ressource 2 dans le module applicatif 1.

10.10 Journal des commandes du serveur SafeKit

Il existe un journal des commandes exécutées sur le serveur SafeKit. Ce journal permet d'effectuer un audit des actions réalisées sur le serveur pour aider au support par exemple. Il enregistre toutes les commandes `safekit` qui sont exécutées sur le serveur et qui modifient le système telles que l'installation d'un module, sa configuration, son lancement/arrêt, le lancement/arrêt du service web de SafeKit, …

Le journal des commandes est stocké dans le fichier `SAFEVAR/log.db` au format SQLite3. Pour lire son contenu :

- exécuter la commande `safekit cmdlog` sur le serveur SafeKit

ou

- cliquer sur l'onglet le journal de commandes depuis la console web.

Ci-dessous un extrait du contenu « brut » du journal de commandes :

```
| 2021-07-27 14:37:33.205122 |   safekit |   mirror |  6883 | START | config -m mirror
| 2021-07-27 14:37:33.400513 |   cluster |   mirror |     0 |     I | update cluster state
| 2021-07-27 14:37:33.405597 |   cluster |   mirror |     0 |     I | module state change on node centos7-
test3
| 2021-07-27 14:37:34.193280 |           |          |  6883 |   END | 0
| 2021-07-27 14:37:34.718292 |   cluster |   mirror |     0 |     I | update cluster state
| 2021-07-27 14:37:34.722080 |   cluster |   mirror |     0 |     I | module state change on node centos7-
test4
| 2021-07-27 14:37:37.510971 |           |          |  6871 |   END | 0
| 2021-07-27 14:38:05.092924 |   safekit |   mirror |  7017 | START | prim -m mirror -u
admin@10.0.0.103
| 2021-07-27 14:38:05.109368 |           |          |  7017 |   END | 0
```

Chaque champ a la signification suivante :

- le 1er correspond à la date d'écriture de l'entrée dans le journal
- le suivant correspond au type d'action exécutée.
- le suivant porte le nom du module si l'action s'applique à un module en particulier
- le suivant contient le pid du processus exécutant l'action
- le suivant vaut `START` au lancement de la commande, suivi du contenu de la commande ; ou bien, il vaut `END` lorsque la commande s'est terminée suivi du code de retour.

10.11 Messages SafeKit dans le journal système

Depuis SafeKit 8, les messages de log des modules SafeKit sont aussi envoyés vers le journal système. Pour les consulter :

- en Windows, ouvrir une console PowerShell et exécuter

```
Get-EventLog -Logname Application -Source Evidian.SafeKit
```

```
  47086 Nov 23 11:27  Information Evidian.SafeKit       1073873154 mirror | heart | Remote state
UNKNOWN Unknown...
  47085 Nov 23 11:27  Information Evidian.SafeKit       1073873154 mirror | heart | Resource
heartbeat.flow set to down by heart...
  47084 Nov 23 11:26  Information Evidian.SafeKit       1073873154 mirror | heart | Local state
ALONE Ready...
  47082 Nov 23 11:26  Warning     Evidian.SafeKit       2147614977 mirror | heartplug | Action
alone called by heart : remote stop...
  47081 Nov 23 11:25  Information Evidian.SafeKit       1073873154 mirror | heart | Remote state
PRIM Ready...
  47080 Nov 23 11:25  Information Evidian.SafeKit       1073873154 mirror | heart | Local state
SECOND Ready...
  47079 Nov 23 11:25  Information Evidian.SafeKit       1073873154 mirror | rfsplug |
Reintegration ended (default)...
```

- en Linux, ouvrir une console Shell et exécuter

```
journalctl -r -t safekit
```

```
Nov 23 15:22:43 localhost.localdomain safekit[3689940]: mirror | heart | Local state ALONE Ready
Nov 23 15:22:43 localhost.localdomain safekit[3689940]: mirror | heart | Local state PRIM Ready
Nov 23 15:16:48 localhost.localdomain safekit[3689940]: mirror | heart | Local state ALONE Ready
```

```
Nov 23 15:16:48 localhost.localdomain safekit[3690096]: mirror | userplug | Script start_prim >
userlog_2023-11-23T151648_start_prim.ulog
Nov 23 15:16:48 localhost.localdomain safekit[3690066]: mirror | rfsplug | Uptodate replicated file
system
Nov 23 15:16:24 localhost.localdomain safekit[3689940]: mirror | heart | Remote state UNKNOWN
Unknown
```

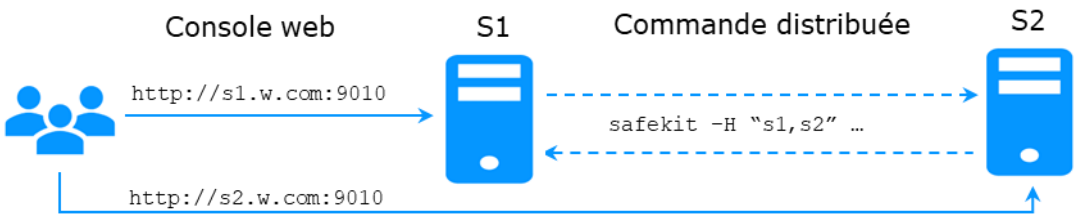
11.Sécurisation du service web de SafeKit

11.1 Vue générale

Le service web de SafeKit est principalement utilisé par :

- La console web (voir section 3)
- L'interface en ligne de commandes distribuées sur le cluster (voir section 9.7)

SafeKit fournit différentes configurations pour ce service afin de renforcer la sécurité de la console web SafeKit et des commandes distribuées.



Protocole	Authentification	Gestion de rôle
✓ HTTP ✓ HTTPS	✓ Aucune (http seulement) ✓ A base de fichiers ✓ LDAP/AD ✓ OpenID Connect	✓ Admin ✓ Control ✓ Monitor

Les configurations les plus sûres sont basées sur HTTPS et l'authentification des utilisateurs.

SafeKit fournit une autorité de certification "privée" (la PKI de SafeKit). Cela permet de sécuriser rapidement SafeKit sans avoir besoin d'une PKI externe (PKI d'entreprise ou PKI commerciale) qui fournit une autorité de certification de confiance.

SafeKit propose également une gestion de rôles basée sur 3 rôles :

Rôle Admin	Ce rôle accorde tous les droits d'administration en autorisant l'accès à Configuration et Supervision dans la barre latérale de navigation
Rôle Control	Ce rôle accorde les droits de contrôle et de supervision en autorisant seulement l'accès à Supervision dans la barre latérale de navigation

Rôle Monitor 	Ce rôle accorde uniquement le droit de supervision en interdisant la possibilité d'actions sur les modules (start, stop…) sous  Supervision dans la barre latérale de navigation

11.1.1 Configuration par défaut

La configuration par défaut est la suivante :

Configuration	Protocole	Authentification/Gestion de rôles
Par défaut	✓ HTTP	✓ Authentification à base de fichiers ✓ Initialisation avec un unique utilisateur `admin`, ayant le rôle Admin ⇨ Pour la configuration, voir section 11.2.1

11.1.2 Configurations prédéfinies

Les configurations prédéfinies sont les suivantes :

Configuration	Protocole	Authentification/Gestion de rôles
Non sécurisée	✓ HTTP	✓ Pas d'authentification ✓ Rôle identique pour tous les utilisateurs Pour faciliter le dépannage ⇨ Pour la configuration, voir section 11.2.2
A base de fichiers	✓ HTTP ✓ HTTPS Pour configurer HTTPS avec : ⇨ la PKI SafeKit, voir 11.3.1 section 185 ⇨ une PKI externe, voir 11.3.2 section 192	✓ Authentification à base de fichiers (nom/mot de passe des utilisateurs stockés dans un fichier Apache) ✓ Gestion de rôles facultative (stockée dans un fichier Apache) ⇨ Pour la configuration, voir section 11.4.1
LDAP/AD	✓ HTTP ✓ HTTPS Pour configurer HTTPS avec : ⇨ la PKI SafeKit, voir section 11.3.1 ⇨ une PKI externe, voir section 11.3.2	✓ Authentification à base de serveur LDAP/AD ✓ Gestion de rôles facultative ⇨ Pour la configuration, voir section 11.4.2

OpenID Connect	✓ HTTPS Pour configurer HTTPS avec : ➪ la PKI SafeKit, voir section 11.3.1 ➪ une PKI externe, voir section 11.3.2	✓ Authentification à base de serveur OpenID Connect ✓ Gestion de rôles facultative ➪ Pour la configuration, voir section 11.4.3

En Linux, pour tous les fichiers ajoutés sous `SAFE/web/conf`, changer leurs droits avec :

```
chown safekit:safekit SAFE/web/conf/<filename>
chmod 0440 SAFE/web/conf/<filename>.
```

11.2 Configuration HTTP

Par défaut, après l'installation de SafeKit, le service web est configuré pour HTTP avec une authentification à base de fichiers qui doit être initialisée.

La configuration par défaut peut être étendue comme décrit en section 11.2.1.

Elle peut aussi être remplacée par la configuration minimale décrite en section 11.2.2 ou une des autres configurations prédéfinies.

11.2.1 Configuration par défaut

La configuration par défaut repose sur HTTP avec une authentification à base de fichiers. Elle nécessite d'être initialisée comme décrit ci-dessous. C'est une étape obligatoire.

Cette configuration par défaut peut être étendue :

- pour rajouter des utilisateurs et leur affecter un rôle, comme décrit en section 11.4.1
- pour passer en HTTPS, avec :
  - ➪ la PKI SafeKit, décrit en 11.3.1
  - ➪ une PKI externe, décrit en section 11.3.2

Après l'installation de SafeKit, la configuration et le redémarrage du service web ne sont pas nécessaires puisqu'il s'agit de la configuration par défaut, et que le service web a été démarré avec celle-ci.

11.2.1.1 Revenir à la configuration HTTP par défaut

Si vous avez modifié la configuration d'authentification utilisateur par défaut et que vous souhaitez revenir à celle-ci, voir la section 11.4.1.

Si vous désirez revenir au mode HTTP, sur tous les serveurs SafeKit :

1. Supprimer le fichier : SAFE/web/conf/ssl/httpd.webconsolessl.conf
2. Exécuter `safekit webserver restart`

(SAFE=C:\safekit en Windows si System Drive=C: ; et SAFE=/opt/safekit en Linux)

11.2.1.2 Initialisation pour la console Web et la commande distribuée

SafeKit fournit un script pour que la console Web et les commandes distribuées soient rapidement opérationnelles.

En Linux, ce script peut être appelé automatiquement lors de l'installation de SafeKit. En Windows, il doit être exécuté manuellement. Dans les deux cas, vous devez spécifier la valeur du mot de passe, *pwd* pour l'utilisateur admin.

SAFE/private/bin/webservercfg -passwd *pwd* où SAFE=C:\safekit en Windows (si %SYSTEMDRIVE%=C:) SAFE=/opt/safekit en Linux	Sur tous les nœuds : 1. Ouvrir une console PowerShell/shell en tant qu'administrateur/root 2. Exécuter la commande SAFE/private/bin/webservercfg -passwd *pwd* *pwd* est la valeur du mot de passe Vous devez affecter le même mot de passe sur tous les nœuds.

Le mot de passe doit être identique sur tous les nœuds du cluster. Dans le cas contraire, la console web et les commandes distribuées échoueront avec des erreurs d'authentification.

Une fois cette initialisation effectuée sur tous les nœuds du cluster :

- Vous pouvez vous authentifier dans la console web avec le nom admin et le mot de passe que vous avez fourni. Le rôle est Admin par défaut (à moins que vous ne changiez le comportement par défaut en fournissant le fichier group.conf comme décrit en section 11.4.1.1).

  En cas d'échec de l'authentification dans la console, vous devrez peut-être réinitialiser le mot de passe. Pour cela, exécutez à nouveau SAFE/private/bin/webservercfg -passwd *pwd* sur tous les nœuds.

- Vous pouvez exécuter des commandes distribuées. Leur authentification est basée sur un utilisateur dédié rcmdadmin avec le rôle Admin. Il est géré dans un fichier utilisateur différent et privé que vous n'avez pas à modifier.

  En cas d'échec de l'authentification pour la commande distribuée, vous devrez peut-être réinitialiser le mot de passe de rcmdadmin. Pour réinitialiser uniquement celui-ci, sans modifier le mot de passe de admin, exécutez SAFE/private/bin/webservercfg -rcmdpasswd *pwd* sur tous les nœuds.

11.2.1.3 Tester la console web et la commande distribuée

La configuration est terminée ; vous pouvez maintenant vérifier qu'elle est opérationnelle :

- Tester la console web

  1. Démarrer un navigateur web

2. Le connecter à l'URL `http://host:9010` (où `host` est l'adresse IP ou le nom d'un nœud SafeKit)
3. Dans la page de connexion, entrer le nom `admin` et le mot de passe spécifié lors de l'initialisation
4. La page chargée autorise toutes les fonctionnalités (rôle Admin par défaut)

- Tester une commande distribuée
  1. Se loguer sur S1 ou S2 en tant que administrateur/root
  2. Ouvrir un terminal (PowerShell, shell, …)
  3. Aller dans le répertoire `SAFE`
  4. Exécuter `safekit -H "*" level`

     qui doit retourner le résultat de la commande `level` sur tous les nœuds

11.2.2 Configuration non sécurisée basée sur un rôle identique pour tous

Elle est basée sur la configuration d'un rôle unique qui est appliqué à tous les utilisateurs sans nécessiter d'authentification. Cette solution ne peut être mise en œuvre qu'en HTTP et est incompatible avec les méthodes d'authentification des utilisateurs. Elle est présente à des fins de dépannage seulement.

11.2.2.1 Configurer et redémarrer le service web

Pour configurer (`SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C: ; et `SAFE=/opt/safekit` en Linux) :

Sur S1 et S2 :

1. éditer le fichier `SAFE/web/conf/httpd.conf`
2. commenter `usefile`, `useldap et useopenid`

```
#Define usefile
…
#Define useldap
…
#Define useopenid
```

3. sélectionner le rôle souhaité en décommentant le port associé et en commentant tous les autres ; si tous les rôles sont commentés, le rôle sélectionné est **Monitor.**

```
Define httpadmin
#Define httpcontrol
```

- `httpadmin` pour le rôle Admin
- `httpcontrol` pour le rôle Control

Sur S1 et S2, désactiver HTTPS s'il avait été active :

4. détruire le fichier `SAFE/web/conf/ssl/httpd.webconsolessl.conf`

	Sur S1 et S2 : 5. exécuter `safekit webserver restart`

11.2.2.2 Tester la console web et la commande distribuée

La configuration est terminée ; vous pouvez maintenant vérifier qu'elle est opérationnelle :

- Tester la console web
  1. Démarrer un navigateur web
  2. Le connecter à l'URL `http://host:9010` (où `host` est l'adresse IP ou le nom d'un nœud SafeKit)
  3. La page chargée donne accès aux fonctionnalités du rôle sélectionné précédemment

- Tester une commande distribuée
  1. Se loguer sur S1 ou S2 en tant que administrateur/root
  2. Ouvrir un terminal (PowerShell, shell, …)
  3. Aller dans le répertoire `SAFE`
  4. Exécuter `safekit -H "*" level`

     qui doit retourner le résultat de la commande `level` sur tous les nœuds

11.3 Configuration HTTPS

Le service web HTTPS s'appuie sur la présence d'un ensemble de certificats énumérés ci-dessous :

	Le certificat de l'Autorité de Certification CA utilisée pour générer les certificats serveur de S1 et S2
	Les certificats serveur de S1 et de S2 permettant de s'assurer de l'identité des nœuds

Appliquez l'une des 2 procédures suivantes pour la configuration HTTPS et des certificats associés :

⇨ Section 11.3.1 « Configuration HTTPS avec la PKI SafeKit »

Aller à cette section pour une configuration rapide de HTTPS avec l'autorité de certification « privée » de SafeKit

⇨ Section 11.3.2 «Configuration HTTPS avec une PKI externe»

Aller à cette section pour configurer HTTPS à l'aide de la PKI externe (PKI d'entreprise ou PKI commerciale) qui fournit une autorité de certification de confiance

A l'issue de cette configuration, vous devez mettre en œuvre une des méthodes d'authentification décrites dans la section 11.4.

11.3.1 Configuration HTTPS avec la PKI SafeKit

Vérifier que l'horloge système est réglée à la date et l'heure courante sur tous les clients et les serveurs. Les certificats portant une date de validité, une différence de date entre les systèmes peut avoir pour effet de les invalider.

11.3.1.1 Choisissez le serveur d'autorité de certification

Tout d'abord, choisissez parmi les nœuds composant le cluster SafeKit, un nœud pour agir en tant que serveur d'autorité de certification. Le nœud sélectionné sera appelé ci-après le `serveur CA`. Les autres nœuds de cluster sont appelés `serveur non-CA`. Appliquez ensuite séquentiellement chacune des sous-sections suivantes pour activer la configuration HTTPS préconfigurée pour sécuriser la console web SafeKit.

11.3.1.2 Démarrer le service web CA sur le serveur CA

Sur le serveur CA :

1. Se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
2. Aller dans le répertoire `SAFE/web/bin`
3. Exécuter la commande `./startcaserv`

   A l'invite, entrer le mot de passe qui protègera l'accès à ce service pour l'utilisateur `CA_admin` (par exemple, `PasW0rD`).

Dans les prochaines étapes, ce mot de passe devra être fourni pour se connecter à ce service.

Le service web CA qui s'exécute sur le `premier serveur`, est aussi accédé par les serveurs supplémentaires non-CA.

Etant donné que ce service écoute sur le port TCP 9001, s'assurer que ce port n'est pas déjà utilisé et qu'il n'est pas bloqué par la configuration du pare-feu. En Linux, le port 9001 est automatiquement ouvert par la commande `startcaserv`. En Windows, la commande `SAFE/private/bin/firewallcfg add` ouvre les communications pour le service `safeacaserv`.

11.3.1.3 Générer les certificats sur le serveur CA

Pendant cette étape, l'environnement de génération des certificats est mis en place ; les certificats de l'autorité de certification et du serveur CA sont générés et installés à l'emplacement attendu par la configuration HTTPS.

Sur le serveur CA :

1. Se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
2. Aller dans le répertoire `SAFE/web/bin`
3. Répertorier les noms DNS et adresses IP du serveur

   Par défaut, le certificat serveur inclut toutes les adresses IP et les noms DNS définis localement. Ils sont répertoriés dans les fichiers `SAFE/web/conf/ipv4.json`,

SAFE/web/conf/ipv6.json et SAFE/web/conf/ipnames.json. Pour générer ces fichiers, exécutez la commande :

- en Linux

  ./getipandnames

  Cette commande utilise sur la commande host délivrée avec le package bind-utils. Installez-le si nécessaire ou remplissez manuellement les noms DNS dans le fichier SAFE/web/conf/ipnames.json.

- en Windows

  ./getipandnames.ps1

Si vous souhaitez accéder à la console web depuis un nom DNS ou une adresse IP non répertorié, modifiez le fichier correspondant pour insérer la nouvelle valeur avant d'exécuter la commande initssl. Cela est nécessaire par exemple pour accéder depuis internet à un cluster SafeKit dans le cloud, quand les serveurs ont une adresse publique mappée sur une adresse privée.

4. Exécuter la commande :

   ./initssl sca

   Cette commande :

   - Crée le certificat CA conf/ca/certs/cacert.crt et de la clé associée conf/ca/private/cacert.key
   - Crée le certificat du serveur conf/ca/certs/server_<HOSTNAME>.crt et de la clé associée conf/ca/private/server_<HOSTNAME>.key
   - Copie le certificat du CA, le certificat serveur et la clé du certificat serveur dans le répertoire conf

Cette commande crée un certificat CA avec un « subject name » par défaut (« SafeKit Local Certificate Authority »). Pour spécifier sa valeur, exécuter plutôt la commande étendue :

```
./initssl sca "/O=My Company/OU=My Entity/CN=My Company Private Certificate Authority".
```

11.3.1.4 Générer les certificats sur le serveur non-CA

Pendant cette étape, le certificat du serveur local est généré, les certificats signés sont téléchargés depuis le serveur CA, et enfin les certificats sont installés à l'emplacement prévu par la configuration HTTPS.

Appliquer en séquence la procédure suivante sur chaque serveur non-CA :

1. Se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
2. Aller dans le répertoire SAFE/web/bin
3. Répertorier les noms DNS et adresses IP du serveur

   Par défaut, le certificat serveur inclut toutes les adresses IP et les noms DNS définis localement. Ils sont répertoriés dans les fichiers SAFE/web/conf/ipv4.json, SAFE/web/conf/ipv6.json et SAFE/web/conf/ipnames.json. Pour générer ces fichiers, exécutez la commande :

   - en Linux

./getipandnames

Cette commande utilise sur la commande `host` délivrée avec le package `bind-utils`. Installez-le si nécessaire ou remplissez manuellement les noms DNS dans le fichier `SAFE/web/conf/ipnames.json`.

- en Windows

  ./getipandnames.ps1

Si vous souhaitez accéder à la console web depuis un nom DNS ou une adresse IP non répertorié, modifiez le fichier correspondant pour insérer la nouvelle valeur avant d'exécuter la commande `initssl`. Cela est nécessaire par exemple pour accéder depuis internet à un cluster SafeKit dans le cloud, quand les serveurs ont une adresse publique mappée sur une adresse privée.

4. Exécuter la commande :

   `./initssl req https://CAserverIP:9001 CA_admin`

   (`CAserverIP` est l'adresse IP ou le nom DNS du serveur CA).

   A chaque fois que cela est demandé, entrer le mot de passe qui a été utilisé au moment du démarrage du service web CA du serveur CA (`PasWOrD`). Pour ne pas avoir à saisir le mot de passe, exécuter plutôt la commande :

   `./initssl req https://CAserverIP:9001 CA_admin:PasW0rD`

Si nécessaire, définissez les variables d'environnement `HTTPS_PROXY` and `HTTP_PROXY` avec les valeurs adéquates.

Si la commande retourne l'erreur "Certificate is not yet valid", cela signifie que les horloges des deux serveurs ne sont pas synchronisées. Pour corriger le problème, il faut changer la date et l'heure des serveurs puis réexécuter la commande `initssl`.

11.3.1.5 Configurer le service web en HTTPS sur les serveurs CA et non-CA

Pour activer la configuration HTTPS, sur tous les serveurs SafeKit :

1. copier `SAFE/web/conf/httpd.webconsolessl.conf` dans `SAFE/web/conf/ssl/httpd.webconsolessl.conf`
2. En Linux exécuter :

   ```
   chown safekit:safekit SAFE/web/conf/ssl/httpd.webconsolessl.conf
   chmod 0440 SAFE/web/conf/ssl/httpd.webconsolessl.conf
   ```

3. exécuter `safekit webserver restart`

(Où `SAFE=C:\safekit` en Windows si System Drive=C: ; et `SAFE=/opt/safekit` en Linux).

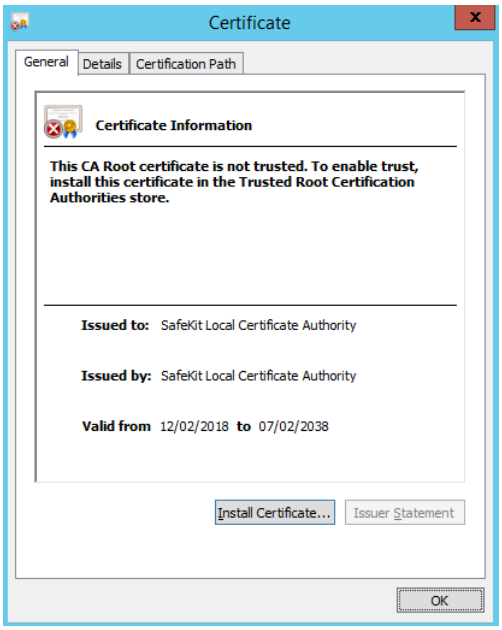
11.3.1.6 Changer les règles du pare-feu sur les serveurs CA et non-CA

Une fois le service Web SafeKit configuré en HTTPS, les communications réseau peuvent être ouvertes en configurant le pare-feu comme décrit en section 10.3.

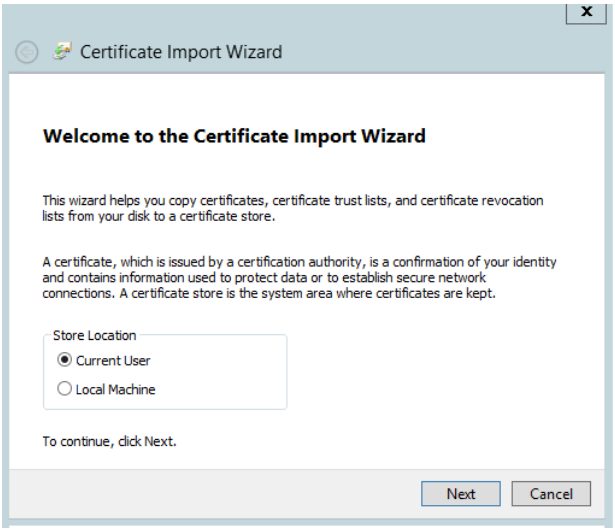
11.3.1.7 Utiliser la console web SafeKit en HTTPS

Tant que le certificat de l'autorité de certification n'a pas été importé, le navigateur émet des alertes de sécurité lorsque l'utilisateur se connecte à la console web avec son certificat client. Si l'importation n'a pas déjà été faite, appliquez la procédure ci-dessous en Windows :

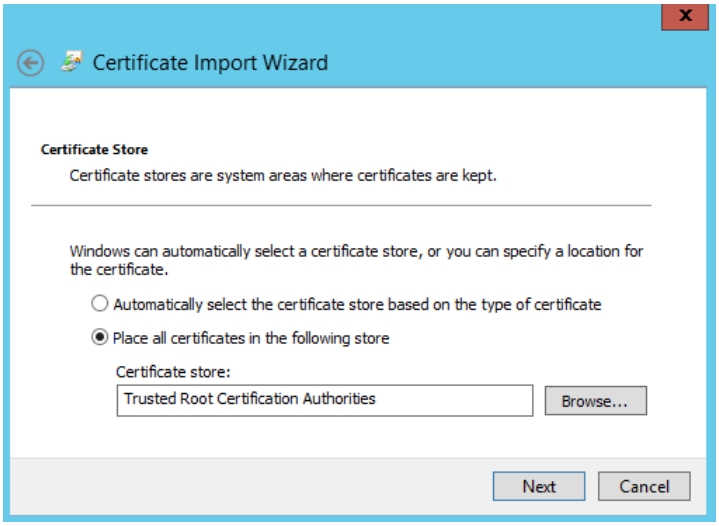
1. Connectez-vous sur la station de travail de l'utilisateur
2. Télécharger depuis le serveur CA le certificat du serveur CA (`cacert.crt` file), localisés dans `SAFE/web/conf/ca/certs`
3. Cliquer sur le fichier `cacert.crt` téléchargé pour ouvrir la fenêtre Certificate. Cliquer ensuite sur le bouton Install Certificate



4. L'assistant d'importation de certificat s'ouvre. Sélectionner Current User
5. Cliquer sur le bouton Next



6. Parcourir les magasins pour sélectionner le magasin Trusted Root Certification Authorities.
7. Cliquer sur le bouton Next



8. Enfin terminer l'importation du certificat

11.3.1.8 Arrêter le service web CA sur le serveur CA

Une fois tous les serveurs configurés, il est recommandé d'arrêter le service web CA (service `safecaserv`) sur le serveur CA. Cela limite le risque d'accès accidentel ou malveillant à l'assistant de configuration HTTPS.

1. Se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
2. Aller dans le répertoire `SAFE/web/bin`
3. Exécuter la commande `./stopcaserv`

En Windows, cette commande supprime également l'entrée du service `safecaserv` pour empêcher son démarrage accidentel par la suite.

En Linux, le port 9001 est automatiquement fermé sur le pare-feu local.

Cette étape n'est pas obligatoire, mais en production il est préférable de ne pas laisser accessible les fichiers sensibles.

Les fichiers présents sur le serveur CA, dans le répertoire `SAFE/web/conf/ca` (en particulier les clés privées sous `SAFE/web/conf/ca/private/*.keys`) doivent être sauvegardés dans un espace de stockage sûr et détruits sur le serveur. Ces fichiers devront être restaurés à leur emplacement initial si le serveur CA est à nouveau nécessaire (par exemple pour sécuriser un nouveau serveur SafeKit).

Pour les serveurs non-CA, il faut sauvegarder et détruire les fichiers présents sous le répertoire `SAFE/web/conf/ca`.

11.3.1.9 Configuration avancée avec la PKI SafeKit

11.3.1.9.1 Renouvellement des certificats

Chaque certificat possède une date d'expiration. Par défaut, la date d'expiration du certificat CA est fixée à 20 ans après la date d'installation. Par défaut, la date d'expiration des certificats serveur est fixée à 20 ans après la date de demande de certificat.

Lorsque le serveur est expiré, la console web émet une alerte lors de la connexion au serveur. Une fois le certificat CA expiré, il sera impossible pour le service web SafeKit de valider les certificats présentés.

Il est possible de renouveler les certificats ou de régénérer une requête de création de certificat à partir des clés privées utilisées précédemment. Cette procédure n'est pas explicitée dans ce document. Il est proposé à la place de créer de nouveau jeu de certificats, pour remplacer les anciens :

1. Supprimer le répertoire `web/conf/ca` sur tous les serveurs, y compris le serveur CA
2. Supprimer les certificats des magasins des stations de travail des utilisateurs
3. Réappliquer complètement les procédures décrites en section 11.3

11.3.1.9.2 Révocation des certificats

Il est possible de modifier la configuration des serveurs web SafeKit pour utiliser une liste de révocation de certificats (CRL). Cette procédure n'est pas explicitée dans ce document. Reportez-vous à la documentation apache et openssl.

Vous pouvez sinon créer un nouveau jeu de certificats, y compris celui de l'autorité de certification, pour remplacer le précédent. Cela a pour effet de révoquer les anciens certificats, car le certificat de l'autorité de certification a changé.

11.3.1.9.3 Commandes pour la génération des certificats

Les commandes doivent être exécutées depuis le répertoire `SAFE/web/bin`.

Tous les chemins d'accès ci-dessous sont relatifs au répertoire SAFE/web.

initssl sca [<subject>]

Paramètres

`<Subject>` : le sujet du certificat du CA, qui identifie le propriétaire du CA.

Exemples

```
initssl sca "/O=My Company/OU=My Unit/CN=My Company Private Certificate
Authority"
```

Description

Cette commande :

- Crée le certificat CA `conf/ca/certs/cacert.crt` et la clé associée `conf/ca/private/cacert.key`
- Crée le certificat du serveur `conf/ca/certs/server_<HOSTNAME>.crt` et la clé associée `conf/ca/private/server_<HOSTNAME>.key`
- Copie le certificat du CA, le certificat serveur et la clé du certificat serveur dans le répertoire conf

Cela initialise le répertoire conf/ca pour les besoins de la PKI SafeKit.

Habituellement, il est préférable de protéger les clés privées par un mot de passe. Cela impliquant une configuration plus complexe, ce n'est pas mis en place. Si besoin, voir la documentation d'Apache et d'OpenSSL.

initssl rca

Comme *initssl sca*, mais réutilise l'autorité de certification préexistante pour regénérer le certificat serveur et sa clé privée, puis installe le certificat de l'autorité de certification, le certificat serveur, et la clé du certificat serveur dans le répertoire conf.

initssl req <url> <user>[:<password>]]

Paramètres

- <url>: Url du service web du serveur CA (https://CAserveur:9001)
- <user>,<password>: utilisateur et mot de passe protégeant l'accès au service web.

La valeur par défaut pour <user> est CA_admin. La valeur pour <password> doit être celle affectée au moment du lancement du service web. Si ce champ n'est pas donné en argument, le script demandera sa saisie.

Exemple

```
initssl req https://CAserveur:9001 CA_admin:PasW0rD
```

Description

Cette commande :

- Crée une requête de certificat pour la génération du certificat serveur. La requête contient toutes les adresses IP et noms DNS associés au serveur local. La requête de certificat est stockée dans conf/ca/private/server_<hostname>.csr et la clé associée dans conf/ca/private/server_<hostname>.key.
- Crée une requête de certificat pour la génération du certificat client avec le rôle Admin (nécessaire pour les commandes distribuées sur le cluster). La requête de certificat est stockée dans conf/ca/private/user_Admin_<hostname>.csr et la clé associée dans conf/ca/private/user_Admin_<hostname>.key.
- Télécharge certificat du CA depuis le serveur CA
- Télécharge depuis le serveur CA, des certificats signés qui ont été générés à partir des requêtes construites précédemment
- Installe les certificats et des clés dans le répertoire conf
- Contrôle de validité des certificats

Si <url> n'est pas spécifié, la commande se termine après avoir généré les requêtes de certificats pour :

- le serveur local (conf/ca/private/server_<hostname>.csr)
- le certificat client avec le rôle Admin (conf/ca/private/user_Admin_<hostname>.csr)

Ces requêtes sont stockées dans le format base64 pour pouvoir être transmises à un CA externe tel Microsoft Active Directory Certificate Services (voir la documentation de Microsoft).

makeusercert <name> <role>

Paramètres

<name> correspond au champ CN du sujet du certificat, habituellement le nom de l'utilisateur du sujet

<role> correspond au rôle lors de l'utilisation de la console web. Les valeurs valides sont Admin ou Control ou Monitor.

Exemples

```
makeusercert administrator Admin
```

```
makeusercert manager Control
makeusercert operator Monitor
```

Description

Création d'une requête de certificat client (ainsi que le certificat, le fichier pkcs12, et la clé associée si la commande est exécutée sur le serveur CA) pour `<name>` et `<role>`.

Lors de la génération du fichier pkcs12, le script demande d'entrer deux fois le mot de passe qui sera utilisé pour protéger son accès. La clé privée non cryptée est stockée dans `conf/ca/private/user_<role>_<name>.key` file. Quand ils ont générés, le certificat est stocké dans `conf/ca/certs/user_<role>_<name>.crt` et le pkcs12 dans `conf/ca/private/user_<role>_<name>.p12`.

Les certificats clients peuvent être utilisés pour authentifier le client lorsqu'il se connecte au service web en HTTPS. Pour pouvoir connecter la console web, le certificat client correspondant au rôle désiré doit d'abord être importé dans le magasin des certificats du navigateur web.

11.3.1.9.4 Service web du serveur de CA

La configuration du service web du serveur de CA se trouve dans le fichier `SAFE/web/conf/httpd.caserv.conf`.

Ce service implémente une sous-partie des fonctionnalités d'un PKI :

- Le certificat CA est accessible depuis l'URL `https://CAserverIP>:9001/<certificate name>.crt`

  Le téléchargement de ce fichier ne nécessite pas de s'authentifier.
- Les requêtes de certificats sont soumises par l'envoi d'un POST à l'URL `https://<CA server IP>:9001/caserv`

  Les arguments sont :

  action = signrequest

  name = <certificate name>

  servercsr = <file content of the server certificate request>

  or

  usercsr = <file content of the client certificate request>

11.3.2 Configuration HTTPS avec une PKI externe

Appliquez les étapes ci-dessous pour configurer HTTPS avec votre autorité de certification de confiance (PKI d'entreprise ou commerciale).

11.3.2.1 Récupérer et installer les certificats serveur

11.3.2.1.1 Récupérer les fichiers certificat

Vous devez récupérer les certificats depuis la PKI dans le format décrit ci-dessous.

	Les certificats serveur de S1 et S2 doivent être signés par l'autorité de certification CA

	Le certificat serveur de S1 permettant de l'authentifier
	Le certificat serveur de S2 permettant de l'authentifier
s1.crt s2.crt	Fichier pour le certificat X509 dans le format PEM Le sous-champ `CN` (`Common Name`) du champ `Objet` (`Subject`) ou le champ `Autre nom de l'objet` (`Subject Alternative Name`), doit contenir : • noms et/ou adresses IP de S1 pour `s1.crt` • noms et/ou adresses IP de S2 pour `s2.crt` Voir l'exemple dans la section 11.3.2.1.3. Attention : vous devez fournir tous les noms et/ou adresses IP utilisés pour la connexion HTTPS : • Celles inclues dans le fichier de configuration du cluster SafeKit • Celles utilisées dans l'URL du navigateur pour charger la console web depuis un nœud du cluster et qui ne sont pas présentes dans la configuration du cluster.
s1.key s2.key	La clé privée, *non cryptée*, associée au certificat `s1.crt` et `s2.crt`

11.3.2.1.2 Installer les fichiers dans SafeKit

Installer les certificats comme suit (`SAFE=C:\safekit` en Windows si System Drive=C: ; et `SAFE=/opt/safekit` en Linux) :

 s1.crt s1.key	Sur S1 : 1. copier `s1.crt` dans `SAFE/web/conf/server.crt` 2. copier `s1.key` dans `SAFE/web/conf/server.key`
 s2.crt s2.key	Sur S2 : 3. copier `s2.crt` dans `SAFE/web/conf/server.crt` 4. copier `s2.key` dans `SAFE/web/conf/server.key`

5. En Linux, sur S1 et S2, exécuter :

```
chown safekit:safekit SAFE/web/conf/server.crt SAFE/web/conf/server.key
```

```
chmod 0440 SAFE/web/conf/server.crt SAFE/web/conf/server.key
```

Vous pouvez contrôler les certificats installés sur le nœud SafeKit avec :

```
cd SAFE/web/bin
checkcert -t server
```

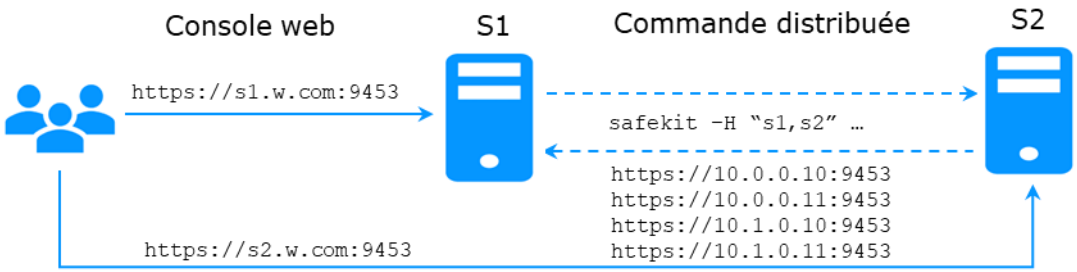
Cette commande retourne en échec si une erreur est détectée.

Vous pouvez également vérifier que le certificat contient bien un nom DNS ou une adresse IP :

```
checkcert -h "nom DNS"
checkcert -i "adresse IP"
```

11.3.2.1.3 Exemple

Prenons comme exemple l'architecture suivante :



Le fichier de configuration pour le cluster SafeKit, `SAFEVAR/cluster/cluster.xml`, contient les valeurs suivantes pour le champ `addr` :

```
<?xml version="1.0"?>
<cluster>
<lans>
  <lan name="default">
    <node name="s1" addr="10.0.0.10"/>
    <node name="s2" addr="10.0.0.11"/>
  </lan>
  <lan name="private">
    <node name="s1" addr="10.1.0.10"/>
    <node name="s2" addr="10.1.0.11"/>
  </lan>
</lans>
</cluster>
```

Le certificat serveur et la configuration du cluster doivent contenir la même liste de valeurs (noms DNS et/ou adresses IP) et les valeurs utilisées pour connecter la console web. Si ce n'est pas le cas, la console web SafeKit et les commandes distribuées ne fonctionneront pas correctement.

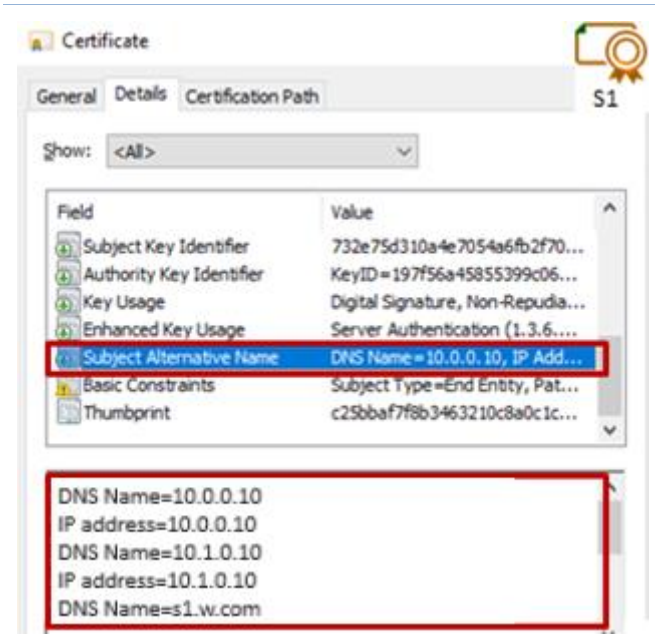
Pour vérifier que cela est bien le cas :

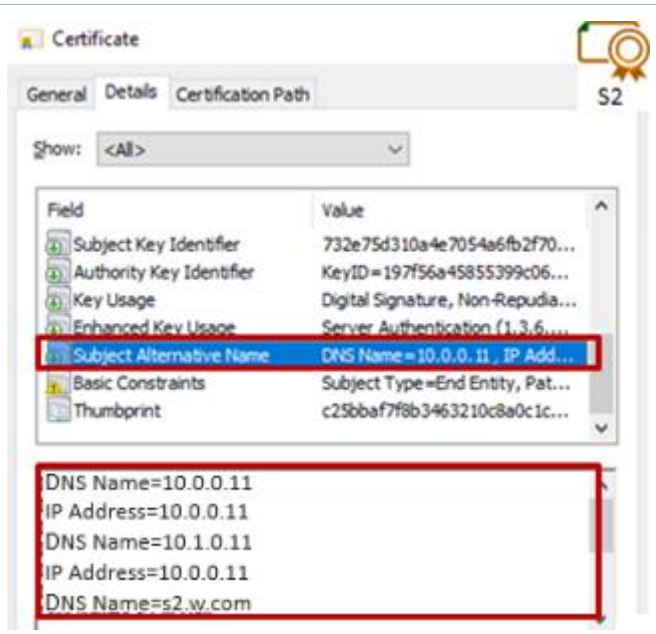
1. Copier le fichier `.crt` (ou `.cer`) sur une station de travail Windows
2. Double cliquer sur le fichier pour l'ouvrir avec Extension noyau de chiffrement
3. Cliquer sur l'onglet Détails
4. Vérifier le contenu du champ `Autre nom de l'objet (Subject Alternative Name)`

Si vous préférez utiliser la ligne de commande, exécutez sur chaque nœud :

```
SAFE/web/bin/openssl.exe x509 -text -noout -in
SAFE/web/conf/server.crt
```

Et vérifier le contenu de la valeur après `Subject Alternative Name`.





11.3.2.2 Récupérer et installer le certificat CA

11.3.2.2.1 Récupérer le fichier du certificat

Vous devez récupérer le certificat de l'Autorité de Certification CA (chaîne de certificats pour la CA racine et les intermédiaires, s'il y en a) utilisé pour générer les certificats serveur de S1 et S2. Le format attendu est le suivant :

 `cacert.crt`	Le certificat de l'Autorité de Certification CA utilisée pour générer les certificats serveur. Fichier pour le certificat X509 dans le format PEM La chaîne de certificats pour la CA racine et les intermédiaires, s'il y en a	 Certificats serveur pour S1 et S2

Si vous rencontrez des difficultés pour récupérer ce fichier depuis la PKI, vous pouvez le construire à l'aide de la procédure décrite en section 7.18.

11.3.2.2.2 Installer le fichier dans SafeKit

Installer le certificat comme suit (`SAFE=C:\safekit` en Windows si System Drive=C: ; et `SAFE=/opt/safekit` en Linux) :

 `cacert.crt`	Sur S1 et S2 : 1. copier `cacert.crt` dans `SAFE/web/conf/cacert.crt` 2. en Linux, exécuter :

	chown safekit:safekit SAFE/web/conf/cacert.crt chmod 0440 SAFE/web/conf/cacert.crt

Vous pouvez contrôler l'installation avec :

```
cd SAFE/web/bin
checkcert -t CA
```

Cette commande retourne en échec si une erreur est détectée.

Vous devez également vérifier que le fichiers `cacert.crt` contient bien la chaîne de certificats pour les autorités de certification racine et intermédiaires.

11.3.2.3 Configurer et redémarrer le service web HTTPS

Pour activer la configuration HTTPS, sur tout les serveurs:

1. copier `SAFE/web/conf/httpd.webconsolessl.conf` dans `SAFE/web/conf/ssl/httpd.webconsolessl.conf`
2. En Linux exécuter :

   `chown safekit:safekit SAFE/web/conf/ssl/httpd.webconsolessl.conf`

   `chmod 0440 SAFE/web/conf/ssl/httpd.webconsolessl.conf`
3. exécuter `safekit webserver restart`

(`SAFE=C:\safekit` en Windows si System Drive=C: ; et `SAFE=/opt/safekit` en Linux)

11.3.2.4 Changer les règles du pare-feu

Vous pouvez exécuter la commande `firewallcfg` pour appliquer les règles pour SafeKit au pare-feu du système d'exploitation (en Windows, `Microsoft Windows Firewall` ; en Linux, `firewalld` ou `iptables`).

Pare-feu	Sur tous les nœuds SafeKit : 1. exécuter `SAFE/private/bin/firewallcfg add` où `SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C: ; et `SAFE=/opt/safekit` en Linux

N'exécutez pas cette commande si vous souhaitez configurer vous-même le pare-feu ou si vous utilisez un pare-feu différent de celui du système. Pour la liste des processus et ports de SafeKit, voir la section 10.3.

11.4 Configuration de l'authentification utilisateur

Mettez en œuvre l'une des méthodes suivantes pour l'authentification utilisateur :

- Section 11.4.1 « Configuration l'authentification à base de fichier »
- Section 11.4.2 « Configuration de l'authentification à base de serveur LDAP/AD »
- Section 11.4.3 « Configuration de l'authentification à base de serveur OpenID Connect »

A l'issue de cette configuration, vous pouvez utiliser la console web sécurisée.

11.4.1 Configuration l'authentification à base de fichier

L'authentification à base de fichier peut être appliquée en HTTP ou HTTPS. Elle repose sur les fichiers suivants :

	Contrôle d'accès à partir du fichier des utilisateurs
	Configuration facultative pour restreindre le rôle de l'utilisateur. Si le fichier `group.conf` n'est pas présent, tous les utilisateurs authentifiés auront le rôle Admin.

11.4.1.1 Gérer les utilisateurs et groupes

Les utilisateurs et groupes doivent être identiques sur S1 et S2, ainsi que les mots de passe. Ils sont définis par les fichiers `user.conf` et `group.conf` dans le répertoire `SAFE/web/conf` (`SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C: ; et `SAFE=/opt/safekit` en Linux).

Pendant l'initialisation de la configuration par défaut, décrite dans la section 11.2.1, l'utilisateur nommé `admin` a été créé et est donc présent dans `user.conf`. Vous pouvez décider de le supprimer si vous en créez d'autres.

1. Création d'un utilisateur

   Les utilisateurs sont créés avec la commande `SAFE/web/bin/htpasswd`.

   Par exemple, pour ajouter le nouvel utilisateur `manager` et lui affecter son mot de passe à `managerpassword`, exécutez :

   ```
   SAFE/web/bin/htpasswd -bB SAFE/web/conf/user.conf manager managerpassword
   ```

   Le nouvel utilisateur est inséré dans le fichier `SAFE/web/conf/user.conf` :


```
admin:$2y$05$oPquL6Z2Y78QcXpHIako.O58Z6lWfa5A86XD.eCbEnbRcguJln9Ce
manager:$apr1$U2GLivF5$x39WKmSpq6BGmLybESgNV1
operator1:$apr1$DetdwaZz$hy5pQzpUlPny3qsXrIS/z1
operator2:$apr1$ICiZv2ru$wRkc3BclBhXzc/4llofoc1
```

2. Affecter un rôle au nouvel utilisateur

   Par défaut, tous les utilisateurs ont le rôle Admin. Si vous souhaitez affecter des rôles différents en fonction des utilisateurs, vous devez créer le fichier `SAFE/web/conf/group.conf` et y définir le rôle de chaque utilisateur. Ce fichier peut contenir les 3 groupes : Admin, Control, Monitor. Les utilisateurs auront le rôle correspondant au groupe auquel ils appartiennent.

   Chaque ligne débute par le nom du groupe, suivi de :, suivi de la liste des utilisateurs (dont le séparateur est l'espace). Voir l'exemple ci-dessous.

   Par exemple, pour affecter le rôle Control au nouvel utilisateur `manager` :

 group.conf	`Admin : admin` `Control : **manager**` `Monitor : operator1 operator2`

Si vous activez la gestion de rôle, vous devez insérer l'utilisateur admin dans `group.conf`. Sinon, cet utilisateur ne sera plus opérationnel.

3. Supprimer un utilisateur, …

   Exécuter `htpasswd -?` Pour lister toutes les options de gestion des utilisateurs.

11.4.1.2 Installer les fichiers

Installer les fichiers comme décrit ci-dessous (`SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C: ; et `SAFE=/opt/safekit` en Linux):

 user.conf	Sur S1 et S2 : 1. copier `user.conf` dans `SAFE/web/conf/user.conf`
 group.conf	Sur S1 et S2, si les groupes sont définis : 2. copier `group.conf` dans `SAFE/web/conf/group.conf`
	3. En Linux, sur S1 et S2, exécuter : `chown safekit:safekit SAFE/web/conf/user.conf SAFE/web/conf/group.conf` `chmod 0440 SAFE/web/conf/user.conf SAFE/web/conf/group.conf`

Ces fichiers doivent identiques sur tous les nœuds.

11.4.1.3 Configurer et redémarrer le service web

Pour activer l'authentification à base de fichier (`SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C: ; et `SAFE=/opt/safekit` en Linux) :

 httpd.conf	Sur S1 et S2 : 1. éditer le fichier `SAFE/web/conf/httpd.conf` 2. si nécessaire, décommenter `usefile` `Define usefile`
	Sur S1 et S2 : 3. exécuter `safekit webserver restart`

11.4.1.4 Tester la console web et la commande distribuée

La configuration est terminée ; vous pouvez maintenant vérifier qu'elle est opérationnelle :

- Tester la console web
  1. Démarrer un navigateur web
  2. Le connecter à l'URL `http://host:9010` (où `host` est l'adresse IP ou le nom d'un nœud SafeKit). Si HTTPS est configuré, il y a une redirection automatique sur `https://host:9453`.
  3. Dans la page de login, entrer le nom d'utilisateur et le mot de passe.

     Avec la configuration par défaut, vous pouvez vous connecter avec l'utilisateur `admin` en donnant le mot de passe que vous lui avez attribué lors de l'initialisation.
  4. La page chargée ne permet que les accès autorisés en fonction du rôle de l'utilisateur. Si les groupes n'ont pas été définis, tous les utilisateurs ont le rôle Admin.
- Tester une commande distribuée
  1. Se loguer sur S1 ou S2 en tant que administrateur/root
  2. Ouvrir un terminal (PowerShell, shell, …)
  3. Aller dans le répertoire `SAFE`
  4. Exécuter `safekit -H "*" level`

     qui doit retourner le résultat de la commande `level` sur tous les nœuds

11.4.2 Configuration de l'authentification à base de serveur LDAP/AD

L'authentification LDAP/AD peut être appliquée en HTTP ou HTTPS. Elle repose sur :

	Contrôle d'accès à partir d'un compte LDAP/AD associé à l'utilisateur
	Configuration facultative des groupes LDAP/AD pour restreindre le rôle de l'utilisateur. Lorsque les groupes ne sont pas définis, tous les utilisateurs authentifiés ont le rôle Admin.

Sur certaines distributions Linux (telles que RedHat 8 et CentOS 8), le démarrage du service web échoue lorsqu'il est configuré avec l'authentification LDAP/AD. Dans ce cas, appliquer la solution décrite dans SK-0092.

Appliquer les étapes décrites ci-dessous après avoir vérifié que S1 et S2 peuvent bien se connecter au port du domaine contrôleur LDAP (par défaut est 389).

11.4.2.1 Créer les utilisateurs et groupes

Si nécessaire, demandez à l'administrateur LDAP de créer les utilisateurs de la console web SafeKit.

Si vous souhaitez restreindre les accès en fonction des rôles, demandez à l'administrateur LDAP de créer les groupes Admin, Control, Monitor et d'affecter les utilisateurs au groupe adéquate. Si les groupes ne sont pas définis, tous les utilisateurs auront le rôle Admin.

11.4.2.2 Configurer et redémarrer le service web

Pour activer l'authentification LDAP/AD (`SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C: ; et `SAFE=/opt/safekit` en Linux) :

Sur S1 et S2 :

Initialiser l'authentification pour la commande distribuée. Cela peut avoir déjà été fait si vous avez initialisé la configuration par défaut après l'installation de SafeKit. Sinon :

1. exécuter `SAFE/private/bin/webservercfg -rcmdpasswd` *pwd*

   où est le *pwd* est le mot de passe pour l'utilisateur privé `rcmdadmin`. Vous n'avez pas besoin de le mémoriser.

Sur S1 et S2 :

2. éditer le fichier `SAFE/web/conf/httpd.conf`
3. décommenter `useldap`

```
Define useldap
```

4. décommenter les lignes suivantes et remplacer les valeurs en gras par celles correspondant à la configuration de votre service LDAP/AD :

```
Define binddn "CN=bindCN,OU=bindOU1,OU=bindOU2,DC=domain,
DC=fq,DC=dn"

Define bindpwd "Password0"

Define searchurl
"ldap://ldaporad.fq.dn:389/OU=searchou,DC=domain,DC=fq,DC=dn
?sAMAccountName, memberOf?sub?(objectClass=*)"
```

les variables `binddn` et `bindpwd` doivent contenir les identifiants d'un compte possédant les droits de recherche dans le répertoire LDAP

la variable `searchurl` définit l'url de recherche (au sens RFC2255) permettant d'authentifier l'utilisateur

`CN`: common name

`OU`: organization unit

`DC`: domain component (one field for each part of the FQDN).

Si aucun groupe n'est défini, tous les utilisateurs authentifiés auront le rôle Admin.

Sur S1 et S2 :

Pour activer la gestion de groupes :

5. éditer le fichier `SAFE/web/conf/httpd.conf`
6. décommenter les lignes suivantes et remplacer les valeurs en gras par celles correspondant à la configuration de votre service LDAP/AD :

```
Define admingroup
"CN=Group1CN,OU=Group1OU1,OU=Group1OU2,DC=domain,DC=fq,DC=dn"
```

```
Define controlgroup
"CN=Group2CN,OU=Group2OU1,OU=Group2OU2,DC=domain,DC=fq,DC=dn"

Define monitorgroup
"CN=Group3CN,OU=Group3OU1,OU=Group3OU2,DC=domain,DC=fq,DC=dn"
```

Les utilisateurs appartenant au groupe `admingroup`, `controlgroup` ou `monitorgroup`, auront respectivement les rôles Admin, Control et Monitor.

Pour une configuration plus avancée, voir la documentation du service web Apache (voir http://httpd.apache.org).

Sur S1 et S2 :

7. exécuter `safekit webserver restart`

11.4.2.3 Tester la console web et la commande distribuée

La configuration est terminée ; vous pouvez maintenant vérifier qu'elle est opérationnelle :

- Tester la console web
  1. Démarrer un navigateur web
  2. Le connecter à l'URL `http://host:9010` (où `host` est l'adresse IP ou le nom d'un nœud SafeKit). Si HTTPS est configuré, il y a une redirection automatique sur `https://host:9453`.
  3. Dans la page de connexion, entrer le nom de l'utilisateur et son mot de passe
  4. La page chargée ne permet que les accès autorisés en fonction du rôle de l'utilisateur. Si les groupes n'ont pas été configurés, tous les utilisateurs ont le rôle Admin.
- Tester une commande distribuée
  1. Se loguer sur S1 ou S2 en tant que administrateur/root
  2. Ouvrir un terminal (PowerShell, shell, …)
  3. Aller dans le répertoire `SAFE`
  4. Exécuter `safekit -H "*" level`

     qui doit retourner le résultat de la commande `level` sur tous les nœuds

11.4.3 Configuration de l'authentification à base de serveur OpenID Connect

Depuis SafeKit 8.2.3, l'authentification à base de serveur OpenID Connect fonctionne uniquement avec HTTPS. Pour configurer HTTPS, voir la section 11.3.

L'authentification OpenID connect s'appuie sur le module Apache `mod_auth_openidc`. Elle doit être appliquée en HTTPS. Elle repose sur :

	Contrôle d'accès à partir d'un compte OpenID associé à l'utilisateur et de l'enregistrement d'une application cliente auprès du fournisseur d'identité OpenID.
	Configuration facultative des attributs utilisateurs pour restreindre le rôle de l'utilisateur. Lorsque les attributs ne sont pas définis, tous les utilisateurs authentifiés ont le rôle Admin.

Sur certaines distributions Linux il peut être nécessaire d'installer le module `mod_auth_openidc`.

Appliquer les étapes décrites ci-dessous après avoir vérifié que S1 et S2 peuvent bien se connecter au fournisseur d'identité OpenID Connect. Il peut être nécessaire de spécifier la configuration d'un mandataire, cf. la section correspondante dans httpd.conf ainsi que la documentation de `mod_auth_openidc` pour plus de détails.

11.4.3.1 Créer les utilisateurs et groupes

Si nécessaire, demandez à l'administrateur OpenID de créer les utilisateurs de la console web SafeKit.

Demander à l'administrateur OpenID d'enregistrer l'app console web safekit et noter les valeurs des identifiants (Client ID et Client Secret) ; ces valeurs seront nécessaires pour effectuer la configuration du serveur web ci-après.

Affecter l'Uri de redirection de l'app à la https://host:9453/openid. S'il est nécessaire de se connecter à plusieurs serveurs, entrer la liste des urls correspondantes.

Si vous souhaitez restreindre les accès en fonction des rôles, demandez à l'administrateur OpenID de créer les groupes ou rôles OpenID Admin, Control, Monitor et d'affecter les utilisateurs au groupe ou rôle OpenID adéquat, puis renseignez les variables `AdminClaim`, `ControlClaim` et `MonitorClaim` avec les valeurs correspondantes dans le fichier httpd.conf. Si les groupes ne sont pas définis, tous les utilisateurs auront le rôle Admin.

Il est également possible de définir les rôles au niveau des serveurs web en affectant des utilisateurs à des rôles dans le fichier `group.conf` comme dans le cas de l'authentification par fichier.

11.4.3.2 Configurer et redémarrer le service web

Pour activer l'authentification OpenID Connect (`SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C: ; et `SAFE=/opt/safekit` en Linux) :

	Sur S1 et S2 : Initialiser l'authentification pour la commande distribuée. Cela peut avoir déjà été fait si vous avez initialisé la configuration par défaut après l'installation de SafeKit. Sinon : 1. exécuter `SAFE/private/bin/webservercfg -rcmdpasswd` *pwd* où est le *pwd* est le mot de passe pour l'utilisateur privé `rcmdadmin`. Vous n'avez pas besoin de le mémoriser.

Sur S1 et S2 :

2. éditer le fichier `SAFE/web/conf/httpd.conf`
3. décommenter `useopenid`

```
Define useopenid
```

4. Localisez les lignes suivantes et remplacez les valeurs correspondant à votre fournisseur d'identité OpenID Connect:

```
OIDCProviderMetadataURL <Your OpenId provider metadata URL>
OIDCClientID <Your OpenID client ID>
OIDCClientSecret <Your OpenID client secret>
OIDCRemoteUserClaim <The Claim in ID token that identifies the
user, if not set, defaults to sub>
## openid connect scope request; this defines which claims are
returned by the IDP.
OIDCScope "openid email"
```

Les variables `OIDCClientID` et `OIDCClientSecret` doivent contenir les identifiants obtenus lors de l'enregistrement de l'application auprès du fournisseur d'identité OpenID Connect.

La variable `OIDCScope` définit les périmètres nécessaires pour obtenir l'attribut définit par `OIDCRemoteUserClaim`, et optionnellement les attributs définissant les rôles. `openid` doit toujours être spécifié.

Si aucune des variables `AdminClaim`, `ControlClaim` et `MonitorClaim` n'est définie, tous les utilisateurs authentifiés auront le rôle Admin.

Sur S1 et S2 :

Pour activer la gestion de groupes :

5. éditer le fichier `SAFE/web/conf/httpd.conf`
6. décommenter les lignes suivantes et remplacer les valeurs en gras par celles correspondant à la configuration de votre service OpenID Connect :

```
# Define AdminClaim roles:SKAdmin
# Define ControlClaim roles:SKControl
# Define MonitorClaim roles:SKMonitor
```

Les utilisateurs portant les attributs `AdminClaim`, `ControlClaim` ou `MonitorClaim` auront respectivement les rôles Admin, Control et Monitor.

Pour une configuration plus avancée, voir la documentation du module `mod_auth_openidc` (voir GitHub - OpenIDC/mod_auth_openidc: OpenID Certified™ OpenID Connect Relying Party implementation for Apache HTTP Server 2.x).

Sur S1 et S2 :

7. exécuter `safekit webserver restart`

11.4.3.3 Tester la console web et la commande distribuée

La configuration est terminée ; vous pouvez maintenant vérifier qu'elle est opérationnelle :

- Tester la console web
  1. Démarrer un navigateur web
  2. Le connecter à l'URL `http://host:9010` (où `host` est l'adresse IP ou le nom d'un nœud SafeKit). Comme HTTPS doit être configuré, il y a une redirection automatique sur `https://host:9453`.
  3. Dans la page de connexion, entrer le nom de l'utilisateur et son mot de passe
  4. La page chargée ne permet que les accès autorisés en fonction du rôle de l'utilisateur. Si les groupes n'ont pas été configurés, tous les utilisateurs ont le rôle Admin.
- Tester une commande distribuée
  1. Se loguer sur S1 ou S2 en tant que administrateur/root
  2. Ouvrir un terminal (PowerShell, shell, …)
  3. Aller dans le répertoire `SAFE`
  4. Exécuter `safekit -H "*" level`

     qui doit retourner le résultat de la commande `level` sur tous les nœuds

12. Cluster.xml pour la configuration du cluster SafeKit

SafeKit utilise le fichier de configuration `cluster.xml`. Ce fichier définit tous les serveurs qui composent le cluster SafeKit ainsi que l'adresse IP (ou nom) de ces serveurs sur les réseaux utilisés pour communiquer avec les nœuds du cluster Il s'agit des communications globales au cluster et internes aux modules ; ces communications étant encryptées. Ce réseau est aussi utilisé pour l'exécution la commande globale `safekit` (avec l'argument `-H`).

Vous devez définir au moins un réseau qui inclut tous les nœuds du cluster. Il est recommandé de définir plusieurs réseaux de type pour tolérer au moins une défaillance réseau.

Le cluster peut être configuré :

- soit via l'assistant de configuration du cluster de la console web SafeKit
- soit en éditant directement le fichier de configuration du cluster et en appliquant la configuration en ligne de commandes

Ces deux méthodes sont décrites en section 12.2.

Pour des exemples complets de configuration voir la section 15.1.1 et la section 15.2.1. Elles présentent la configuration via la console web ainsi que le `cluster.xml` correspondant.

12.1 Le fichier `cluster.xml`

Chaque réseau (`lan`) possède un nom logique qui sera utilisé dans la configuration des modules pour nommer les réseaux de surveillance :

- dans la section `heartbeat` pour un module miroir (voir la section 13.3)
- dans la section `lan` pour un module ferme (voir la section 13.4)

Le nom du nœud est utilisé par le service d'administration de SafeKit (`safeadmin`) pour identifier de manière unique un nœud SafeKit. Vous devez toujours utiliser le même nom pour désigner le même serveur sur les différents réseaux. Ce nom est aussi utilisé par la console web SafeKit lors de l'affichage du nom du nœud.

12.1.1 Cluster.xml exemple

- Dans l'exemple ci-dessous, 2 réseaux sont définis. L'un des réseaux peut être dédié au flux de la réplication de fichiers pour un module miroir.

```
<cluster>
   <lans>
      <lan name="default">
         <node name="node1" addr="192.168.1.67"/>
         <node name="node2" addr="192.168.1.68"/>
      </lan>
      <lan name="repli">
         <node name="node1" addr="10.0.0.1"/>
```

```
        <node name="node2" addr="10.0.0.2"/>
      </lan>
   </lans>
</cluster>
```

- Dans l'exemple ci-dessous, un seul réseau est utilisé, mais dans une configuration NAT (Network address translation). Pour chaque nœud du cluster deux adresses doivent être définies : l'adresse locale `laddr` (celle de l'interface locale) et l'adresse externe `addr` (celle connue des autres nœuds).

```
<cluster>
   <lans>
      <lan name="default">
         <node name="node1" addr="server1.dns.name" laddr="10.0.0.1"/>
         <node name="node2" addr="server2.dns.name" laddr="10.0.0.2"/>
      </lan>
   </lans>
</cluster>
```

Tous les nœuds doivent pouvoir communiquer avec les autres via les adresses externes.

12.1.2 Cluster.xml syntaxe

```
<cluster>
  <lans [port="4800"]>
    <lan name="lan_name" [command="on|off"] >
       <node name="node_name" addr="IP1_address"|"IP1_name"
             [ laddr="local_IP1_address" ]/>
       <node name="node_name" addr="IP2_address"|"IP2_name"
             [ laddr="local_IP2_address" ]/>
       …
    </lan>
    …
  </lans>
</cluster>
```

12.1.3 <lans>, <lan>, <node> attributs

`<lans`	Début de la définition des nœuds du cluster et de la topologie réseau
`[port="xxxx"]`	Port UDP sur lequel le protocole `membership` échange. Valeur par défaut : `4800`
`[pulse="xxxx"]`	Période d'émission des messages du protocole d'appartenance. Un pulse élevé utilise moins de bande passante, mais entraîne un délai de réaction plus long.
`[mlost_count="xx"]`	Nombre de périodes écoulées sans réception de message avant d'élire un nouveau leader.
`[slost_count="xx"]`	Nombre de périodes écoulées sans réception de message avant de déclarer un follower hors ligne.

<lan	Définition d'un réseau sur lequel s'exécute le protocole membership. Au moins un réseau. Autant de sections qu'il y a de LANs entre les serveurs.
name="*lan name*"	Nom logique unique pour le réseau Ce nom est utilisé dans la configuration du module pour définir les réseaux utilisés.
command="on"|"off"	Mettre command="on" pour utiliser ce réseau pour l'exécution des commandes distribuées sur le cluster. Dans ce cas, la section <lan> doit inclure tous les nœuds qui composent le cluster. Il faut définir une unique section <lan> avec command="on". Quand cet attribut n'est pas positionné, c'est la première section <lan> qui est utilisée pour l'exécution des commandes distribuées sur le cluster. Valeur par défaut : off
<node	Définition d'un nœud dans le réseau. Positionner autant de nœuds qu'il y a de serveurs dans le cluster (au moins 2).
name="*node name*"	Nom unique logique pour le serveur SafeKit Vous devez toujours utiliser le même nom pour désigner le même serveur sur les différents réseaux.
addr= "*IP_address*"| "*IP_name*"	Adresse IPv4 ou IPv6, ou nom du nœud tel qu'il est connu par les autres nœuds dans le LAN (préférer une adresse IP pour être indépendant d'un serveur de noms DNS). Pour des configurations NAT, c'est l'adresse extérieure qui doit être indiquée. Lors de la définition d'une adresse IPv6, utiliser le format littéral : l'adresse est placée entre crochets (par exemple [2001::7334])
laddr= "*local_IP_address*"	Adresse IP locale dans le LAN. A utiliser uniquement pour les configurations NAT.

Dans SafeKit < 8.2, la configuration du cluster avait des attributs console et framework sur la balise <lan>. Ces attributs étaient nécessaires pour l'ancienne console web et sont obsolètes avec la nouvelle. S'ils sont présents, ces attributs sont ignorés en SafeKit 8.2..

12.2 Configuration du cluster SafeKit

12.2.1 Configuration avec la console web de SafeKit

La console web de SafeKit fournit un assistant de configuration pour éditer le fichier `cluster.xml` et appliquer la configuration sur tous les nœuds qui composent le cluster.

- La configuration du cluster nécessite de se connecter à la console web avec un utilisateur ayant le rôle Admin

- Si le cluster n'est pas encore configuré, la console web ouvre automatiquement l'« Assistant de configuration du cluster »
- Quand le cluster est configuré, la configuration courante du cluster est chargée depuis le nœud de connexion spécifiée dans l'URL du navigateur

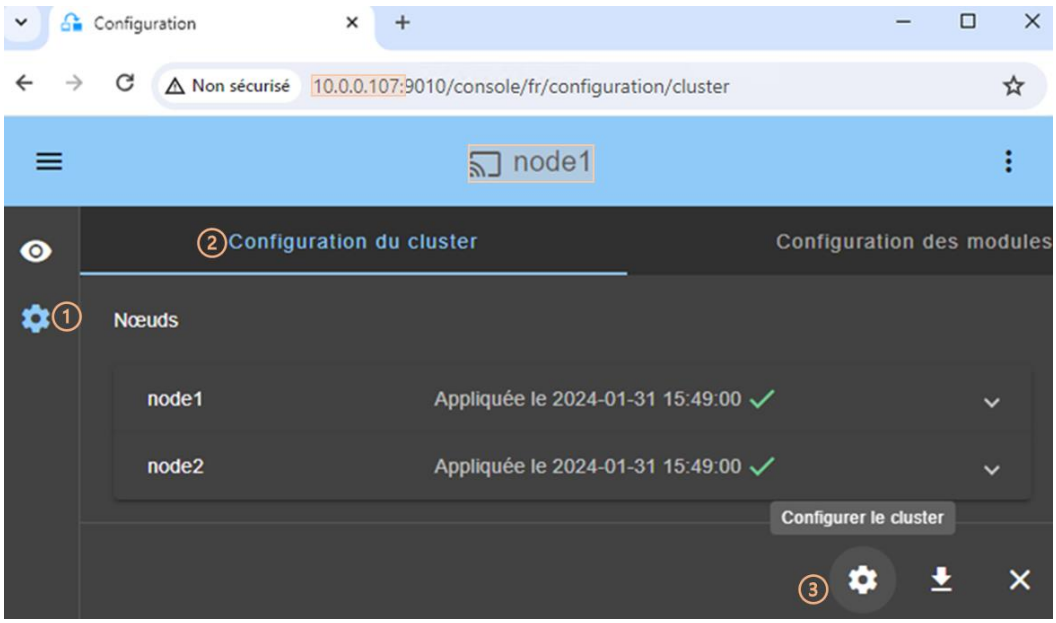
Ouvrir l'assistant de configuration du cluster

- Directement via l'URL http://host:9010/console/fr/configuration/cluster/config

Ou

- En naviguant dans la console

Dans cet exemple, la console est chargée depuis `10.0.0.107` qui correspond au nœud `node1` dans le cluster existant.



- (1) Cliquer sur ⚙« Configuration » dans la barre de navigation latérale
- (2) Cliquer sur l'onglet « Configuration du cluster »
- (3) Cliquer sur le bouton ⚙« Configurer le cluster »

Pour des détails sur l'assistant de configuration du cluster, voir la section 3.2.1.

12.2.2 Configuration en ligne de commande

- (1) Se loguer en tant que administrateur/root

- (2) Éditer le fichier `SAFEVAR/cluster/cluster.xml`

  sur Windows, `SAFEVAR= C:\safekit\var` si `%SYSTEMDRIVE%`=C:

  sur Linux, `SAFEVAR=/var/safekit`

- (3) Appliquer la configuration du cluster avec une nouvelle clé de chiffrement en exécutant :

  1. `safekit cluster config`

     configure localement, depuis le fichier `cluster.xml`, et génère une nouvelle la clé de chiffrement pour la communication entre les nœuds

  2. `safekit -H "*" -G`

     la configuration local, définie dans le fichier `cluster.xml`, est appliquée sur tous les nœuds du cluster

Pour reconfigurer le cluster sans clé de chiffrement :

1. `safekit cluster delkey`
2. `safekit -H "*" -G`

Pour régénérer des clés de chiffrement et les propager sur les nœuds du cluster :

1. `safekit cluster genkey`
2. `safekit -H "*" -G`

Pour la description complète des commandes, se référer à la section 9.2.

12.2.3 Changements de configuration

Lorsque la configuration du cluster SafeKit est modifiée, la nouvelle configuration doit impérativement être appliquée sur tous les serveurs qui composent le cluster. Si la configuration n'est appliquée que sur un sous-ensemble des nœuds présents dans la configuration du cluster, seul le sous-ensemble des nœuds sera en mesure de communiquer. Cela peut avoir pour conséquence de perturber le fonctionnement des modules installés sur les serveurs. Pour rétablir un fonctionnement correct, vous devez réappliquer la configuration sur tous les nœuds du cluster comme décrit précédemment.

Il est possible d'afficher la configuration courante en exécutant la commande `safekit cluster confinfo` sur chaque nœud (voir section 9.2). Quand la configuration est correcte, cette commande retourne sur tous les nœuds, la même liste de nœuds et la même signature de configuration.

Changer la configuration du cluster peut aussi avoir un impact important sur la configuration des modules car les noms logiques de réseau `<lan>` sont utilisés dans les configurations de modules. Tout changement de configuration déclenche une mise à jour des modules démarrés pour prendre en compte ces modifications. Cela peut conduire à un arrêt de ces modules en cas d'incompatibilité (par exemple sur destruction d'un réseau alors qu'il est utilisé par un module). Aussi, il faut être prudent lors de la modification de la configuration du cluster quand des modules sont en cours de fonctionnement.

13. Userconfig.xml pour la configuration du module

Chaque fois que vous modifiez `userconfig.xml`, la configuration doit être appliquée à tous les nœuds du cluster sur lesquels le module est installé pour être prise en compte.

Pour appliquer la nouvelle configuration, modifiée sur `node1`, sur tous les nœuds, suivre la procédure suivante (remplacer *node1*, *node2* par le nom des nœuds et *AM* par le nom du module) :

- la console web, connectée à `node1`, en naviguant sur ⚙ « Configuration/Configuration des modules/⚙Configurer le module »
- ou en entrant directement l'URL http://*node1*:9010/console/en/configuration/modules/*AM*/config/
- ou avec la commande `safekit config -H "node1,node2" -E AM` exécutée sur *node1*

Exemple de `userconfig.xml` :

```
<safe>
  <!-- Insert below <macro> <service> tags -->
</safe>
```

Avec la console web, le module doit être arrêté avant d'appliquer la configuration.

Avec la commande, il est possible d'appliquer la configuration alors que le module est démarré, mais uniquement dans l'état ✓`ALONE (Ready)` et ○`WAIT (NotReady)`. Cette fonctionnalité est appelée *configuration dynamiq*ue. Uniquement un sous-ensemble restreint de la configuration peut être modifié dynamiquement. Quand la nouvelle configuration ne peut être appliquée, un message d'erreur est affiché. Les attributs de configuration pouvant être changés dynamiquement sont listés ci-après.

13.1 Macro définition - <macro>

Utiliser une macro pour associer un nom à une valeur. Dans le `userconfig.xml`, le nom, encadré par le caractère `%`, est remplacé par la valeur de la macro portant ce nom.

La syntaxe `%identifier%` peut être aussi utilisée dans `userconfig.xml` pour récupérer la valeur d'une variable d'environnement de nom `identifier`. En cas de conflit, c'est la valeur de macro qui prime.

13.1.1 <macro> Exemple

Dans l'exemple suivant, `%PATH%` est remplacé par `e:\path`.

```
<macro name="PATH" value="e:\path"/>
<service>
  …
  <rfs>
   <replicated dir="%PATH%" />
  </rfs>
</service
```

Un exemple d'utilisation des macros est donné dans la section 15.3. Il décrit la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.1.2 <macro> Syntaxe

```
<macro
     name="identifier"
     value="value"
/>
```

13.1.3 <macro> Attributs

`<macro`	
`name="identifier"`	Une chaîne de caractères qui identifie la macro.
`value="value"`	La valeur qui remplacera chaque occurrence de `%identifier%` dans la suite de `userconfig.xml`.
`/>`	

13.2 Module ferme ou miroir - <service>

13.2.1 <service> Exemple

- Exemple pour un module miroir

```
<service mode="mirror"
defaultprim="alone" maxloop="3" loop_interval="24" failover="on">
  <!-- Insérer ci-dessous les tags <hearbeat> <rfs> <vip> <user> <vhost>
<errd> <check> <failover> -->
</service>
```

Pour un exemple complet d'un module miroir, voir la section 15.1. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

- Exemple pour un module ferme

```
<service mode="farm" maxloop="3" loop_interval="24">
  <!-- Insérer ci-dessous les tags <farm> <vip> <user> <vhost> <errd> <check>
<failover> -->
</service>
```

Pour un exemple complet d'un module ferme, voir la section 15.2. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant

13.2.2 <service> Syntaxe

```
<service mode="mirror"|"farm"|"light"
  [boot="off"|"on"|"auto"|"ignore"]
  [boot_delay="0"]
  [failover="on"|"off"]
  [defaultprim="alone"|"server_name"|"lastprim"]
  [maxloop="3"] [loop_interval="24"]
  [automatic_reboot="off"|"on"]>
</service>
```

Seuls les attributs `boot`, `maxloop`, `loop_interval` et `automatic_reboot` peuvent être modifiés dynamiquement.

13.2.3 <service> Attributs

`<service`	Première section à définir dans un module
`mode= "mirror"| "farm"| "light"`	• `mode="mirror"` pour un module miroir (voir section 1.2) Le protocole de synchronisation entre les 2 serveurs est défini en section 13.3. Pour un exemple complet d'un module miroir, voir la section 15.1. • `mode="farm"` pour un module ferme (voir section 1.3) Le protocole de synchronisation entre les serveurs est défini en section 13.4.

	 Pour un exemple complet d'un module ferme, voir la section 15.2. • `mode="light"` (voir section 1.2.9) Configuration sur un seul serveur avec uniquement la détection d'erreur et le redémarrage local.
`[boot="on"|"off"|"auto"|"ignore"]`	• `boot="on"` le module est automatiquement démarré au boot de la machine. • `boot="off"` le module n'est pas démarré au boot de la machine. • `boot="auto"` le module est démarré automatiquement au boot de la machine s'il était démarré avant le reboot. • `boot="ignore"` Avant SafeKit 7.5, la configuration du démarrage au boot du module se faisait avec la commande `safekit boot -m` *AM* `on|off` (qui devait être exécutée sur chaque nœud). Si vous préférez continuer à utiliser cette commande, supprimez l'attribut `boot` ou affectez-lui la valeur `ignore` (valeur par défaut). Le module ne sera pas démarré au boot, à moins que la commande `safekit boot -m` *AM* `on` ne soit exécutée. L'état de la configuration du démarrage au boot est visible dans la ressource `usersetting.boot`. L'état des ressources est visible dans la console web/ Contrôle /Sélection du nœud/onglet Ressources ; via la commande `safekit state -m` *AM* `-v` Valeur par défaut : `off`
`[boot_delay="0"]`	Le délai, en secondes, avant le démarrage du module au boot. Valeur par défaut : `0` (pas de délai)
`[failover="on"|"off"]`	Pour un module miroir seulement. • `failover="on"` reprise automatique sur le serveur secondaire lorsque le serveur primaire est arrêté. • `failover="off"` lorsque le serveur primaire est arrêté, le serveur secondaire se met en attente (pas de reprise automatique). Seule la commande `safekit prim` peut forcer le démarrage du serveur secondaire en primaire. Pour une description, voir la section 5.7. Valeur par défaut : `on`

[defaultprim= "alone"| "*server_name*"| "lastprim"]	Pour un module miroir seulement. `defaultprim` décide quel serveur parmi 2 serveurs est le serveur primaire par défaut pour un module applicatif. Cette option est utile quand un module est `ALONE` sur un serveur et que le module est démarré sur l'autre serveur. • `defaultprim="alone"` le module `ALONE` devient `PRIM` alors que le module redémarré devient `SECOND`. Valeur recommandée pour éviter les basculements d'application juste après la réintégration. • `defaultprim="server_name"` lorsque le module tourne sur deux serveurs, le serveur `PRIM` est celui indiqué dans `defaultprim`. Cette valeur peut être utile dans les architectures actif/actif (voir section 1.4.2) ou N-1 (voir section 1.4.3). • `defaultprim="lastprim"`, le serveur redémarré redevient `PRIM` s'il était `PRIM` avant son dernier arrêt. Valeur par défaut : `alone`
[maxloop="3"]	Nombre de détections d'erreur successives avant arrêt. Cet attribut définit le nombre maximum d'actions (`restart`, `stopstart`, `wait`) pouvant être exécutées, à la suite d'une détection d'erreur émise par `<errd>` ou un `<checker>`, avant d'arrêter localement le module. Ce compteur est réinitialisé à l'expiration du délai `loop_interval` ou lors d'une commande manuelle `safekit start, restart, swap, stopstart…`. Noter qu'une commande `safekit` émise par un détecteur avec l'option `-i identity` incrémente le compteur, alors qu'une commande manuelle sans cette option le réinitialise. La valeur de cet attribut peut être modifiée dynamiquement. `maxloop` est représenté par la ressource `heart.stopstartloop`. Sa valeur courante correspond à la date à laquelle le compteur a été initialisé (sous la forme d'un timestamp Epoch Unix) ; et sa date d'affectation correspond soit à son initialisation, soit à un rebouclage (`stopstart`, `restart`). Consulter l'historique des ressources pour visualiser chaque incrémentation du compteur. Valeur par défaut : `3`
[loop_interval ="24"]	Intervalle de temps, en heures, sur lequel `maxloop` s'applique. Affectez sa valeur à 0 pour désactiver le compteur `maxloop`. Valeur par défaut : `24` heures

	La valeur de cet attribut peut être modifiée dynamiquement.
[automatic_reboot ="off"| "on"]	Si positionné à on, reboot sur safekit stopstart au lieu de stopper puis redémarrer. Valeur par défaut : off La valeur de cet attribut peut être modifiée dynamiquement.

13.3 Heartbeats - <heart>, <heartbeat >

Les heartbeats doivent être utilisés seulement avec les modules miroirs. La topologie d'une ferme est décrite dans la section 13.4.

Le mécanisme basique pour synchroniser deux serveurs et détecter les défaillances d'un serveur est le heartbeat (battement de cœur), qui consiste en un échange de petits paquets UDP entre les serveurs. Normalement, on met autant de heartbeats qu'il y a de réseaux connectant les 2 serveurs. Dans une situation normale, les 2 serveurs échangent leurs états (PRIM, SECOND, les états des ressources) à travers les heartbeats et synchronisent ainsi les procédures de démarrage arrêt applicatif.

Si tous les heartbeats sont perdus, cela signifie que l'autre serveur est en panne. Le serveur local décide de devenir ALONE. Bien que non obligatoire, il est préférable d'avoir deux voies de heartbeats sur deux réseaux différents afin de distinguer une panne réseau d'une panne serveur et d'éviter le cas du split-brain.

Le réseau utilisé par le heartbeat est défini par le nom logique d'un réseau de la configuration du cluster SafeKit (voir section 12).

13.3.1 <heart> Exemple

Pour un exemple complet d'un module miroir, voir section 15.1. Elle présente la configuration via la console web ainsi que le userconfig.xml correspondant.

- Exemple pour configurer un heartbeat sur le réseau du cluster nommé default

```
<heart>
   <heartbeat name="default" />
</heart>
```

- Exemple avec 2 heartbeats, dont un réseau de réplication dédié, configuré avec ident="flow" sur le réseau du cluster nommé private

```
<heart>
   <heartbeat name="default" />
   <heartbeat name="private" ident="flow"/>
</heart>
```

13.3.2 <heart> Syntaxe

```
<heart
  [port="xxxx"] [pulse="700"] [timeout="30000"]
  [permanent_arp="on"]
>
```

```
 <heartbeat
  [port="xxxx"] [pulse="700"] [timeout="30000"] name="network" [ident="name"]
 >
</hearbeat>
 …
</heart>
```

Le tag <heart> et son sous-arbre peuvent être entièrement modifiés dynamiquement.

13.3.3 <heart>, <heartbeat> Attributs

<heart	
[port="xxxx"]	Port UDP sur lequel les heartbeats sont échangés. Valeur par défaut : dépend de l'id du module applicatif. Rendu par la commande safekit module getports.
[pulse="700"]	Délai en milliseconde entre l'émission de 2 paquets de heartbeat. Valeur par défaut : 700 ms
[timeout="30000"]	Timeout de détection en ms de la perte d'un heartbeat. Valeur par défaut : 30 000 ms
<heartbeat	Définition d'un heartbeart. Il y a autant de sections <heartbeat> qu'il y a de voies de heartbeats (réseaux connectant les serveurs). Au moins 1 heartbeat.
[port="xxxx"]	Redéfinition du port de heartbeat. Par défaut le même que celui dans <heart>.
[pulse="700"]	Redéfinition du délai en milliseconde entre l'émission de 2 paquets de heartbeat. Par défaut le même que celui dans <heart>.
[timeout= "30000"]	Redéfinition du timeout de détection en ms de la perte d'un heartbeat. Par défaut le même que celui dans <heart>.
name="*network*"	Nom du réseau utilisé par le heartbeat. "network" doit être le nom d'un réseau défini dans la configuration du cluster SafeKit (voir section 12).
[ident="*name*"]	Donne le nom qui sera utilisé pour ce heartbeat dans la console web ainsi que pour la ressource interne correspondante, i.e. : heartbeat.name peut être utilisé dans la failover machine (voir section 13.18). Si l'attribut ident n'est pas défini la valeur de l'attribut name est utilisée.

	 Si vous positionnez un heartbeat `ident="flow"`, le flux de réplication sera positionné automatiquement sur la même voie. Si vous positionnez `ident="flow"` sans configuration <rfs>, le démarrage du module bloque dans l'état `WAIT`.
`[permanent_arp= "on"|"off"]`	Régulièrement, `heart` positionne un ARP permanent pour ses voies de heartbeats. Cette procédure peut geler `heart` sur certains systèmes (Linux). Dans ce cas, positionner à "`off`" et affecter l'ARP permanent sur les voies de heartbeats au boot. Sur Linux, ceci peut être fait en ajoutant la commande suivante dans un script exécuté au boot : `arp -s hostname hw_addr` Valeur par défaut : `on`
`<server addr= "IP1_address" />`	**Ancienne** définition de l'adresse ip du serveur pour ce heartbeat Le tag `<server>` était utilisé dans l'ancienne syntaxe de configuration (avant SafeKit 7.2). Il est supporté pour assurer la compatibilité ascendante, mais ne doit pas être utilisé pour la configuration de nouveaux modules.  Vous ne devez pas utiliser dans le même `userconfig.xml`, la syntaxe de SafeKit 7.1 et celle introduite depuis SafeKit 7.2.

13.4 Topologie d'une ferme - <farm>, <lan>

Le mécanisme basique pour synchroniser une ferme de serveurs est un protocole de groupe qui détecte automatiquement les membres disponibles dans le groupe (protocole membership).

Le réseau utilisé par le protocole est défini par le nom logique d'un réseau de la configuration du cluster SafeKit (voir section 12).

13.4.1 <farm> Exemple

Exemple d'utilisation du réseau du cluster nommé `default`.

```
<farm>
   <lan name="default" />
</farm>
```

 Pour des exemples complets de configuration dans un module ferme, voir la section 15.2. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.4.2 <farm> Syntaxe

```
<farm [port="xxxx"]>
   <lan name="network">
   </lan>
```

```
   …
</farm>
```

Le tag <farm> et son sous-arbre **ne peuvent pas** être modifiés dynamiquement.

13.4.3 <farm>, <lan> Attributs

<farm	
[port="xxxx"]	Port UDP sur lequel le protocole membership échange. Valeur par défaut : dépend de l'id du module applicatif. Rendu par la commande safekit module getports
[pulse= "xxx"]	Période d'émission des messages du protocole d'appartenance. Un pulse élevé utilise moins de bande passante, mais entraîne un délai de réaction plus long.
[mlost_count= "xx "]	Nombre de périodes écoulées sans réception de message avant d'élire un nouveau leader.
[slost_count= "xx "]	Nombre de périodes écoulées sans réception de message avant de déclarer un follower hors ligne.
<lan	Définition d'un lan sur lequel s'exécute le protocole membership. Au moins un lan doit être défini. Autant de sections qu'il y a de réseaux entre les serveurs.
name="*network*"	Nom du réseau utilisé. network doit être le nom d'un réseau défini dans la configuration du cluster SafeKit (voir section 12).
< node name="identity" addr= "IP1_address" />	Adresse IP et nom du nœud dans ce lan. Le tag <node> était utilisé dans l'ancienne syntaxe de configuration (avant SafeKit 7.2). Il est supporté pour assurer la compatibilité ascendante, mais ne doit pas être utilisé pour la configuration de nouveaux modules. Vous ne devez pas utiliser dans le même userconfig.xml, la syntaxe de SafeKit 7.1 et celle introduite depuis SafeKit 7.2.

13.5 Adresse IP virtuelle - <vip>

Si vous installez plusieurs modules applicatifs sur le même serveur, l'adresse IP virtuelle doit être différente pour chaque module.

13.5.1 <vip> Exemple dans un module miroir

L'exemple ci-dessous configure l'adresse IP virtuelle sur le nœud primaire d'un cluster sur site :

```
<vip>
```

```
    <interface_list>
        <interface check="on" arpreroute="on">
           <real_interface>
               <virtual_addr addr="192.168.1.222" where="one_side_alias"
check="on"/>
          </real_interface>
        </interface>
    </interface_list>
  </vip>
```

Voir aussi l'exemple complet en section 15.1. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.5.2 <vip> Exemple dans un module ferme

L'exemple ci-dessous configure l'équilibrage de charge, à destination du port 80 et de l'adresse IP virtuelle, entre les nœuds d'un cluster sur site :

```
<vip>
    <interface_list>
      <interface check="on" arpreroute="on" arpinterval="60" arpelapse="10">
        <virtual_interface type="vmac_directed">
          <virtual_addr addr="192.168.1.222" where="alias" check="on"/>
        </virtual_interface>
      </interface>
    </interface_list>
    <loadbalancing_list>
       <group name="FarmProto">
         <rule port="80" proto="tcp" filter="on_port"/>
       </group>
    </loadbalancing_list>
  </vip>
```

Voir aussi l'exemple complet en section 15.2. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.5.3 Alternative à <vip> pour des serveurs dans des réseaux IP différents

La configuration d'une adresse IP virtuelle avec une section `<vip>` dans `userconfig.xml` requiert des serveurs dans le même réseau IP (reroutage réseau et équilibrage de charge effectués au niveau 2).

Si les serveurs se trouvent dans des réseaux IP différents, la section `<vip>` ne peut pas être configurée. Dans ce cas, une alternative consiste à configurer l'adresse IP virtuelle dans un équilibreur de charge. L'équilibreur de charge achemine les paquets vers les adresses IP physiques des serveurs en testant le status d'une URL nommée vérificateur d'état (health check) et géré par SafeKit.

SafeKit fournit donc un vérificateur d'état pour chaque module. Vous devez configurer le test de vérification dans le load balancer avec :

- le protocole HTTP
- le port 9010, port du service web de SafeKit
- l'URL `/var/modules/`*`AM`*`/ready.txt` où *`AM`* est le nom du module

Pour un module miroir, le test retourne :

- `OK`, qui signifie que l'instance est saine, quand le module est dans l'état ✓`PRIM (Ready)` ou ✓`ALONE (Ready)`
- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états

Pour un module ferme, le test retourne :

- `OK`, qui signifie l'instance est saine, quand le module est dans l'état ✓`UP (Ready)`
- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états

Une autre alternative consiste à ce que vous implémentiez une configuration DNS spéciale et une commande de redirection DNS insérée dans les scripts de redémarrage de SafeKit.

13.5.4 <vip> Syntaxe

13.5.4.1 Partage de charge réseau dans une architecture ferme

```
<vip [tcpreset="off"|"on"]>
 <interface_list>
  <interface
    [check="off"|"on"]
    [arpreroute="off"|"on"]
    [arpinterval="60"]
    [arpelapse="1200"]
  >
   <virtual_interface
   [type="vmac_directed"|"vmac_invisible"]
   [addr="xx:xx:xx:xx:xx"]
   >
     <virtual_addr
      addr="virtual_IP_name"|"virtual_IP_address"
      [where="alias"]
      [check="off"|"on"]
      [connections="off"|"on"]
     />
     …
   </virtual_interface>
</interface>
 </interface_list>
 <loadbalancing_list>
   <group name="group_name"
     <cluster>
        <host name="node_name" power="value" />
        …
     </cluster>
     <rule
       [virtual_addr="*"|"virtual_IP_name"|"virtual_IP_address"]
       [port="*"|"value"]
        proto="udp"|"tcp"
        filter="on_addr"|"on_port"|"on_ipid"
     />
     …
   </group>
   …
 </loadbalancing_list>
```

```
</vip>
```

Le tag <vip> et son sous-arbre **ne peuvent pas** être modifiés dynamiquement.

13.5.4.2 Basculement réseau dans une architecture miroir

Pour un cluster sur site :

```
<vip [tcpreset="off"|"on"]>
 <interface_list>
  <interface
    [check="off"|"on"]
    [arpreroute="off"|"on"]
    [arpinterval="60"]
    [arpelapse="1200"]
  >

   <real_interface>
    <virtual_addr
      addr="virtual_IP_name"|"virtual_IP_address"
      where="one_side_alias"
      [check="off"|"on"]
      [connections="off"|"on"]
    />
    …
   </real_interface>
  </interface>
  …
 </interface_list>

</vip>
```

13.5.5 <interface_list>, <interface>, <virtual_interface>, <real_interface>, <virtual_addr> Attributs

<vip>	
[tcpreset="off"|"on"]	Avant de déconfigurer l'adresse IP virtuelle, les connexions l'ayant comme IP source, sont rompues. La rupture de connexion est désactivée en positionnant cet attribut à off. Valeur par défaut : on
<interface_list>	
<interface	Définition des adresses IP virtuelles sur une interface. Mettre autant de sections <interface> que vous avez d'interfaces réseau à configurer.
[check="off"|"on"]	Positionner un checker sur l'interface réseau. Le module est mis dans l'état WAIT tant que l'interface est down. Le nom du checker d'interface est intf.<network_IP_mask> (intf.192.168.0.0). Valeur par défaut : on

	Pour plus d'informations, voir section 13.13.
[arpreroute="off"|"on"]	Broadcast de gratuitous ARP pour le reroutage des adresses IP virtuelles définies dans les sections <real_interface>. Valeur par défaut : off
[arpinterval="60"]	Temps en seconde entre 2 gratuitous ARP. Valeur par défaut : 60 s
[arpelapse="1200"]	Temps total pendant lequel des gratuitous ARP sont émis. Valeur par défaut : 1200 s
[name="*interface name*"]	**Linux seulement** Vous pouvez spécifiez le nom de l'interface. Exemple, positionner name="bond0" Par défaut, SafeKit détecte l'interface réseau à configurer à partir des adresses IP virtuelles configurées sur cette interface.

13.5.5.1 <virtual_interface>, <virtual_addr> Attributs dans une architecture ferme

A utiliser pour les modules ferme avec partage de charge sur l'IP virtuelle :

<virtual_interface	Définition des adresses IP virtuelles configurées sur une interface Ethernet.
type= "vmac_directed"| "vmac_invisible"	• type="vmac_directed" Associe l'adresse MAC de l'un des serveurs à l'adresse IP virtuelle, comme pour le reste du trafic normal. Voir la description en section 13.5.7.3. • type="vmac_invisible" Adresse MAC virtuelle jamais visible dans les entêtes Ethernet pour permettre le broadcast des switchs. Nécessite le support du mode promiscuous. Voir la description en section 13.5.7.2. Lorsque vous exécutez SafeKit dans une machine virtuelle, il faut activer le mode promiscuous sur l'adaptateur réseau virtuel. Voir SK-0099 pour la procédure à suivre pour Hyper-V et VMware ESX. Note : configuration possible pour un module miroir et pour un reroutage transparent sans gratuitous ARP.
[addr="xx:xx:xx:xx:xx"]	Unicast virtual MAC adresse. Si non positionné, par défaut concaténation de "5A:FE" (Safe) et de la 1ère adresse IP virtuelle en hexadécimal.

	Ignoré lorsque `type="vmac_directed"`
`<virtual_addr`	Définition d'une adresse IP virtuelle. Mettre autant de lignes `<virtual_addr>` qu'il y a d'adresses IP virtuelles à configurer sur l'interface.
`addr="`*`virtual_IP_name`*`"|"`*`virtual_IP_address`*`"`	Nom ou adresse IP virtuelle (préférer une adresse IP pour être indépendant de la panne du serveur de nom). Adresse IPv4 ou IPv6.
`where="alias"`	L'adresse IP virtuelle est définie sur tous les serveurs de la ferme en alias. Note : Dans le cas particulier d'une configuration d'un module miroir avec VMAC mettre ici `where="one_side_alias"`.
`[check="off"|"on"]`	Positionner un IP checker sur l'adresse virtuelle. Le module exécute un stopstart quand l'IP virtuelle est détruite. Le nom de l'IP checker est `ip.<virtual addr value>` (`ip.192.168.1.99`). Valeur par défaut : `on` Pour plus d'informations, voir section 13.14.
`[connections="off"|"on"]`	Active le comptage du nombre de connexions actives sur l'adresse virtuelle. Ce nombre est stocké dans la ressource nommée `connections.<addr value>` (par exemple : `connections.192.168.1.99`) qui est affectée toutes les 10 secondes. Cette valeur est fournie à un titre indicatif uniquement. Valeur par défaut : `off`
`netmask="`*`defaultnetmask`*`"`	**Linux et IPV4 seulement** Par défaut, prend le netmask de l'interface. A positionner si l'interface a plusieurs netmasks.
`</virtual_interface>`	

13.5.5.2 <real_interface>, <virtual_addr> Attributs dans une architecture miroir

A utiliser pour les modules miroir avec basculement de l'IP virtuelle :

`<real_interface>`	Définition d'adresses IP virtuelle associée avec l'adresse MAC réelle de l'interface.
`<virtual_addr`	Définition d'une adresse IP virtuelle. Mettre autant de lignes `<virtual_addr>` qu'il y a d'adresses IP virtuelles à configurer sur l'interface.

addr= "*virtual_IP_name*"| "*virtual_IP_address*"	Nom ou adresse IP virtuelle (préférer une adresse IP pour être indépendant de la panne du serveur de nom). Adresse IPv4 ou IPv6.
where="one_side_alias"	Adresse mise en alias sur le serveur PRIM ou ALONE.
[check="off"|"on"]	Positionner un IP checker sur l'adresse virtuelle. Le module exécute un stopstart quand l'IP virtuelle est détruite. Le nom de l'IP checker est ip.<addr value> (ip.192.168.1.99). Valeur par défaut : on Pour plus d'informations, voir section 13.14.
[connections="off"|"on"]	Active le comptage du nombre de connexions actives sur l'adresse virtuelle. Ce nombre est stocké dans la ressource nommée connections.<virtual addr value> (par exemple : connections.192.168.1.99) qui est affectée toutes les 10 secondes. Cette valeur est fournie à un titre indicatif uniquement. Valeur par défaut : off
netmask="*defaultnetmask*"	**Linux et IPV4 seulement** Par défaut, prend le netmask de l'interface. A positionner si l'interface a plusieurs netmasks.
</real_interface>	

13.5.6 <loadbalancing_list>, <group>, <cluster>, <host> Attributs

A utiliser uniquement avec un module ferme.

Voir aussi l'exemple complet en section 15.2. Elle présente la configuration via la console web ainsi que le userconfig.xml correspondant.

<loadbalancing_list>	
<group	Définition d'un groupe de load balancing. Mettre autant de sections qu'il y a de groupes. Voir l'exemple de plusieurs groupes en section 15.2.2.4.
name="*group_name*"	Nom du groupe de load balancing.
<cluster	Définition des serveurs et des poids. Sans la section <cluster>, les règles s'appliquent sur tous les serveurs de la ferme.
<host	Définition d'un nœud dans le groupe

name="*node_name*"	Nom du nœud utilisé. node_name doit être le nom d'un serveur défini dans la configuration du cluster SafeKit (voir section 12).
power="*value*"	Poids du nœud dans le groupe. Peut-être égal à 0 si l'on ne veut aucun trafic sur le nœud. Pour plus d'informations, voir section 13.5.7.4.
</cluster>	
<rule	Définition d'une règle de load balancing dans le groupe. Autant de lignes que de règles de load balancing.
[virtual_addr= "*" | "*virtual_IP_address*" | "*virtual_IP_name*"]	Adresses IP virtuelles concernées par le load balancing. Par défaut toutes : *
[port="*"|"*value*"]	Port TCP ou UDP sur lequel s'applique la règle de load balancing Par défaut tous les ports : *
proto="udp" | "tcp"| "arp"	• proto="udp" règle de load balancing UDP • proto="tcp" règle de load balancing TCP • proto="arp" règle de load balancing pour le protocole de résolution IP<->MAC.
filter="on_addr" | "on_port" | "on_ipid"	• filter="on_addr" La règle de load balancing est réalisée sur l'adresse IP client en entrée. • filter="on_port" La règle de load balancing est réalisée sur le port client en entrée. • filter="on_ipid" La règle de load balancing est réalisée sur l'ip_id en entrée. Utile pour UDP seulement

13.5.7 <vip> Description

13.5.7.1 <vip> prérequis

Voir les prérequis réseau décrits en section 2.3.2.

13.5.7.2 Qu'est-ce que le type "vmac_invisible" ?

La configuration `type="vmac_invisible"` associe une adresse MAC virtuelle à l'adresse IP virtuelle. Avec une adresse MAC virtuelle, les paquets émis vers l'adresse IP virtuelle sont reçus par tous les serveurs. Dans un module noyau, chaque serveur décode le paquet réseau et l'accepte ou le rejette. Après une sélection à très bas niveau des paquets réseau, l'application sur le serveur gère seulement le trafic réseau sélectionné par le module noyau.

Le mécanisme d'adresse MAC virtuelle consiste à associer une adresse MAC unicast dite virtuelle à l'adresse IP virtuelle. Quand un routeur ou une machine réseau recherche l'adresse IP virtuelle, les serveurs SafeKit répondent avec l'adresse MAC virtuelle (via le protocole standard). Cependant, chaque serveur utilise son adresse MAC physique pour communiquer. Ainsi, l'adresse MAC virtuelle est invisible et non localisable par les switchs Ethernet. Par défaut, les switchs émettent ce type de paquet sur tous leurs ports (flooding), ceux-ci sont alors reçus par tous les serveurs de la ferme.

Avec la technologie d'adresse MAC virtuelle, le reroutage en cas de panne et de reprise est immédiat. Tous les équipements conservent l'association adresse IP virtuelle, adresse MAC virtuelle dans leur cache ARP.

Lorsque vous exécutez SafeKit dans une machine virtuelle, il peut être nécessaire d'activer le mode promiscuous sur l'adaptateur réseau virtuel. Voir SK-0099 pour la procédure à suivre pour Hyper-V et VMware ESXi..

Pour tester une adresse MAC virtuelle sur votre réseau, faire d'abord le test de compatibilité décrit en section 4.3.7.

13.5.7.3 Qu'est-ce que le type "vmac_directed" ?

La configuration `type= "vmac_directed"` modifie le fonctionnement du filtre. Dans ce mode, il n'y a pas de MAC virtuelle ; vu de l'extérieur, l'adresse IP virtuelle se comporte comme une adresse IP normale du point de vue de la résolution IP<->MAC.

Le module noyau est chargé de filtrer et transmettre les paquets entrant au serveur désigné par l'algorithme de partage de charge.

Le mode `"vmac_directed"` introduit un délai pour les clients ayant résolu l'adresse IP virtuelle sur l'adresse MAC d'un serveur qui est devenu indisponible. Ceci est comparable à ce qui se passe dans le cas `<real_interface>`. Les autres clients ne sont pas affectés.

Pour minimiser ce délai en IPV4, positionner `arpreroute="on"` sur l'interface correspondante, et régler les paramètres `arpelapse` et `arpinterval`.

Ipv6 possède un mécanisme interne et ne nécessite pas de configuration particulière.

13.5.7.4 Comment fonctionne le load balancing ?

Dans un module kernel, l'algorithme de load balancing est réalisé par filtrage sur les l'identité des paquets en réception. Cette identité est définie par configuration dans `userconfig.xml` : adresse IP client, port client … (i.e. : load balancing de niveau 4). L'identité est passée dans une table de hachage (une bitmap de 256 bits) qui indique si le paquet doit être accepté ou rejeté sur le serveur. Un seul filtre accepte le paquet dans la ferme de serveurs. Quand un serveur est défaillant, le protocole membership reconfigure les filtres pour redistribuer le trafic du serveur défaillant sur les serveurs disponibles.

Chaque serveur peut avoir un poids (=1, 2…) et prendre plus ou moins de trafic. Le poids est mis en œuvre par le nombre bits à 1 dans la table de hachage (la bitmap de 256 bits).


Un exemple de bitmap est donné en section 4.3.5.

13.6 Réplication de fichiers - <rfs>, <replicated>

S'applique à un module miroir seulement.

Sur Linux, vous devez définir la même valeur pour les uid/gid sur les deux nœuds pour la réplication des permissions sur les fichiers. Lors de la réplication d'un point de montage du système de fichiers, vous devez appliquer une procédure spéciale décrite en section 13.6.4.2.

Sur Windows, il est vivement recommandé d'activer le journal USN sur le lecteur contenant le répertoire répliqué, comme décrit en section 13.6.4.3.


Si vous exécutez plusieurs modules simultanément, les répertoires répliqués doivent être différents pour chaque module.

13.6.1 <rfs> Exemple

- Exemple en Windows

```
<rfs async="second" >
      <replicated dir="c:\safedir" mode="read_only"/>
</rfs>
```

- Exemple en Linux

```
<rfs async="second" >
      <replicated dir="/safedir" mode="read_only"/>
</rfs>
```


Voir aussi l'exemple complet en section 15.1. Pour la configuration d'un réseau de réplication dédié, voir la section 15.1.2.2. Elles présentent la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.6.2 <rfs> Syntaxe

```
<rfs
     [acl="on"|"off"]
     [async="second"|"none"]

     [iotimeout="nb seconds"]
     [roflags="0x10"|"0x10000"]
     [locktimeout="100"]
     [sendtimeout="30"]

     [nbrei="3"]
     [ruzone_blocksize="8388608"]
     [namespacepolicy="0"|"1"|"3"|"4"]
     [reitimeout="150"]
     [reicommit="0"]
     [reidetail="on"|"off"]
     [allocthreshold="0"]
```

```
     [nbremconn ="1"]

     [checktime="220000"]
     [checkintv="120"]

     [nfsbox_options="cross"|"nocross"]
     [scripts="off"]
     [reiallowedbw="20000"]
     [syncdelta="nb minutes"]
     [syncat="planification de la synchronisation"]
>
 <replicated dir="absolute path of a directory"
  [mode="read_only"]>
  <tocheck path="relative path of a file or subdir" />
  <notreplicated path="relative path of a file or subdir" />
  <notreplicated regexpath="regular expression on relative path of a file or
subdir" />
  …
 </replicated>
</rfs>
```

Seuls les attributs `async`, `nbrei`, `reitimeout` et `reidetail` du tag `<rfs>` peuvent être modifiés par une configuration dynamique. Le tag `<flow>`, qui décrit le flux de réplication, peut également être changé dynamiquement.

13.6.3 <rfs>, <replicated> Attributs

`<rfs`	
`[mountoversuffix= "suffix"]`	Sur Linux uniquement. À la configuration du module miroir, le répertoire répliqué "`/a/dir`" est renommé en "`/a/dirsuffix`". Le répertoire **/a/dir** est créé et c'est : • un point de montage vers `/a/dirsuffix` lorsque le module est démarré • un lien vers "`/a/dirsuffix`" lorsque le module est arrêté Par défaut le suffix est « _For_SafeKit_Replication » S'il y a une défaillance matérielle, le lien symbolique n'est pas restauré. Dans ce cas, vous devez le restaurer manuellement. **Restriction** Vous ne pouvez pas spécifier directement une racine de système de fichiers comme répertoire répliqué (car le renommage du répertoire racine ne fonctionne pas). Le contournement consiste à une manipulation du fichier `fstab` tel qu'expliquée dans un KB sur https://support.evidian.com.

	 Lorsque le module est démarré, NE PAS ACCEDER les fichiers dans "`/a/dir`*`suffix`*", car les modifications ne seront pas répliquées et le système deviendra incohérent. TOUJOURS ACCEDER les fichiers via "`/a/dir`".
`[acl= "on" | "off"]`	Active la réplication des ACLs sur les fichiers et répertoires. Valeur par défaut : `off` **Restrictions sur Windows**  La réplication des ACLs ne fonctionnera pas si le compte SYSTEM n'a pas les droits "Full control" sur toute la forêt répliquée. Le service `safeadmin` s'exécute dans le compte SYSTEM. Les ACLs sont répliqués littéralement (SID), sans translation, donc les ACLs "local users/groups" ne sont pas utilisables sur le serveur distant. Le cryptage et la compression des fichiers ne sont pas supportés.
`[async= "second" | "none"]`	Positionner `async="second"` améliore les performances de la réplication de fichiers : les opérations d'écriture répliquées sont mises en cache sur le serveur secondaire et les acquittements sont envoyés plus rapidement au serveur primaire. • `async="none"` Cela assure plus de sécurité : les opérations d'écriture répliquées sont mises sur disque avant d'envoyer les acquittements au serveur primaire. • `async="second"` En cas de double panne simultanée des 2 serveurs `PRIM` et `SECOND`, si le serveur `PRIM` ne peut pas redémarrer, alors le serveur `SECOND` n'a pas les données à jour sur son disque. Il y a perte de données si on force le serveur `SECOND` à redémarrer en primaire avec la commande prim. Valeur par défaut : `second`  La valeur de cet attribut peut être modifiée dynamiquement.
`[packetsize]`	**Linux seulement** Taille maximale en octet des paquets de réplication NFS. Elle doit être inférieure ou égale à la taille maximale des paquets supportée par le serveur NFS des 2 serveurs. Quand cet attribut est affecté dans la configuration, il est utilisé pour affecter rsize et wsize au montage NFS. Par défaut, la taille est celle du serveur NFS.

[reipacketsize="8388608"]	Taille maximale en octets des paquets de réintégration. En Linux, cette taille doit être inférieure ou égale à packetsize. Valeur par défaut en Linux : valeur de packetsize si elle est affectée dans la configuration et est < 8388608; sinon 8388608 Valeur par défaut en Windows : 8388608 octets
[ruzone_blocksize="8388608"]	Taille en octet d'une zone pour la bitmap de modification d'un fichier. Ça doit être un multiple de l'attribut reipacketsize. Valeur par défaut : valeur de reipacketsize si elle est affectée dans la configuration ; sinon 8388608
[iotimeout]	**Windows seulement** Timeout en seconde sur les IO gérées dans le filtre file system Windows. Si une IO ne peut pas être répliquée et si le timeout du filtre expire, alors le serveur PRIM devient ALONE. Si non positionné, la valeur par défaut est calculée dynamiquement.
[roflags="0x10"|"0x10000"]	**Windows seulement** • roflags="0x10" Pour garantir la cohérence des données répliquées sur les 2 serveurs, la modification des répertoires/fichiers répliqués ne doit avoir lieu que sur le serveur PRIM. Si des modifications ont lieu sur le serveur SECOND, celles-ci sont notifiées dans le journal du module avec l'identification du processus responsable afin que l'administrateur puisse corriger cette anomalie. C'est le comportement avec roflags="0x10". • roflags="0x10000" Depuis SafeKit 7.4.0.31, le module peut en plus être arrêté sur le serveur SECOND en définissant roflags="0x10000". Valeur par défaut : 0x10
[locktimeout="100"]	Timeout en secondes des requêtes répliquées. Si une requête ne peut pas être traitée dans cet intervalle de temps, le serveur PRIM devient ALONE Valeur par défaut : 100 secondes
[sendtimeout="30"]	Depuis SafeKit> 7.4.0.5 Timeout en secondes pour l'envoi de paquets TCP au nœud distant. Si l'envoi du paquet n'a pas pu être effectué dans le délai imparti, le serveur PRIM devient ALONE. Augmentez cette valeur en cas de réseau lent. Valeur par défaut : 30 secondes

	 Dans SafeKit 7.4.0.5, la valeur par défaut était de 120 secondes.
`[nbrei="3"]`	Nombre de threads de réintégration s'exécutant en parallèle pour resynchroniser les fichiers. Valeur par défaut : `3`  La valeur de cet attribut peut être modifiée dynamiquement.
`[namespacepolicy="0"|"1"|"3"|"4"]`	• `namespacepolicy="0"` pour désactiver la synchronisation par zones sur Windows ou Linux. • `namespacepolicy="1"` En Windows, la réintégration par zones ne peut être assurée après l'arrêt propre du module, puis reboot du serveur. • `namespacepolicy="3"` En Windows, cela permet la réintégration par zones après reboot quand cela est possible. Cette option exploite l'USN journal du volume qui contient le répertoire répliqué (voir la commande `fsutil usn` pour la création du journal). Malgré cette option, la réintégration complète doit être appliquée lorsque : o l'USN journal associé au volume a été détruit/recréé par l'administrateur o une discontinuité dans le journal est détectée • `namespacepolicy = "4"` Lorsque la synchronisation par zones n'est pas possible (lors de la première réintégration ou lorsque les zones ne sont pas disponibles), les fichiers devant être synchronisés sont entièrement copiés. Si cette réintégration ne se termine pas, la suivante copiera à nouveau ces fichiers. Pour éviter cela, définissez `namespacepolicy = "4"`. Cette option active également la vérification de journal USN en Windows. Valeur par défaut : `4` pour SafeKit > 7.4.0.5 (non supporté dans les versions antérieures)
`[reitimeout="150"]`	Timeout en seconde des requêtes de réintégration. Ce timeout peut être augmenté pour éviter les arrêts de réintégration à cause d'une machine primaire chargée. Valeur par défaut : `150` secondes  La valeur de cet attribut peut être modifiée dynamiquement.
`[reicommit="0"]`	**Linux seulement**

	Positionner `reicommit="nb blocks"` pour commiter sur disque tous les (nb blocks)*`reipacketsize` lors de la réintégration d'un fichier (en plus du commit réalisé à la fin de la copie). Ceci peut aider la réintégration de gros fichiers mais ralentit le temps de réintégration global. Valeur par défaut : `0` signifie pas de commit intermédiaire.
`[reidetail= "on"|"off"]`	Journal détaillé de la réintégration. Valeur par défaut : `off`  La valeur de cet attribut peut être modifiée dynamiquement.
`[allocthreshold = "0"]`	**Windows seulement** Taille en Go pour appliquer la politique d'allocation avant réintégration. Quand `allocthreshold > 0`, activation de l'allocation rapide de l'espace disque pour les fichiers à réintégrer sur le nœud secondaire. Cette fonctionnalité permet, quand le fichier est très gros (> 200 Go) et pas encore complètement recopié, d'éviter le timeout de l'écriture du primaire en fin de fichier. Depuis SafeKit 7.4.0.64, la politique d'allocation a changé et est appliquée : • pour les nouveaux fichiers (fichiers n'existant pas sur la secondaire quand la réintégration commence) • quand la taille du fichier sur la primaire est >= `allocthreshold` (taille en Go) • pour une synchronisation de type `full`: o Lors de la 1ère réintégration o Lors d'un démarrage en réintégration complète (`safekit second fullsync`) o Quand la réintégration par zone est désactivée (`namespacepolicy="0"`). Valeur par défaut : `0` (qui désactive la fonctionnalité)
`[nbremconn="1"]`	Nombre de connexions TCP entre les nœuds primaire et secondaire. Cette valeur peut être augmentée pour améliorer le débit de réplication et de synchronisation lorsque le réseau présente une latence élevée (dans le cloud, par exemple). Valeur par défaut : `1`
`[checktime= "220000"]`	**Linux seulement**

	Timeout en millisecondes pour une requête null qui vérifie le bon fonctionnement local de la réplication de fichiers. Commande stopstart si le timeout est atteint. Valeur par défaut : `220000`
`[checkintv= "120"]`	**Linux seulement** Intervalle en secondes entre 2 requêtes null. Valeur par défaut : `120` secondes
`nfsbox_options= "cross"|"nocross"`	**Windows seulement** Cette option spécifie la politique à appliquer lorsqu'un reparse point du type `MOUNT_POINT` est présent dans l'arborescence répliquée. Cette politique est globale à tous les répertoires répliqués. Dans NTFS, les reparse point de type `MOUNT_POINT` représentent : • soit un point de montage NTFS (par exemple `D:\directory`) • soit un "directory junction" NTFS (en quelque sorte un lien symbolique vers une autre partie du système de fichiers) • `nfsbox_options="cross"` les points de montages sont évalués avant réplication et le contenu de la cible du point de montage est répliqué, ce qui rend le point de montage équivalent à un répertoire normal. Ce comportement est utile lorsque le répertoire répliqué est la racine d'un système de fichier (par exemple D:\ ). C'est le comportement par défaut. • `nfsbox_options="nocross"` les points de montages ne sont pas évalués et sont répliqués en tant que point de montage (fichier de type « reparse point »). Le contenu de la cible du point de montage n'est pas répliqué ou réintégré lorsque le point de montage est sollicité. Ce comportement est utile lorsque la cible du point de montage est située dans un autre répertoire répliqué (fichier de type « junctions »). Par exemple, les bases de données PostgreSQL utilisent des fichiers de ce type. Valeur par défaut : `cross`
`[scripts= "on" | "off"]`	`scripts="on"` active les callback vers les scripts `_rfs_*` utilisés pour mettre en œuvre une réplication de données particulière Valeur par défaut : `off`

[reiallowedbw= "20000"]	Quand cet attribut est défini, il spécifie la bande passante maximum susceptible d'être utilisée par la phase de réintégration (par exemple 20000 KB/s), en kilo octets par secondes (KB/s). Etant donné l'implémentation retenue, une fluctuation de +/- 10% de la bande passante réellement utilisée est observable. La bande passante utilisée par la réplication n'est pas affectée par ce paramètre. Par défaut : l'attribut n'est pas défini et la bande passante utilisée par la réintégration n'est pas limitée
[syncdelta="*nb minutes*"]	• `syncdelta <=1` l'attribut est ignoré et la politique par défaut de démarrage et de reprise sur panne est appliquée : seul le serveur avec les données à jour peut démarrer en primaire ou effectuer une reprise sur panne. • `syncdelta >1` la politique par défaut de démarrage et de reprise sur panne est modifiée. Le serveur avec des données non à jour peut devenir primaire mais uniquement si le temps écoulé depuis sa dernière synchronisation est inférieur à la valeur de `syncdelta` (en minutes). Valeur par défaut : `0` minutes
[syncat="*planification de la synchronisation*"]	Valeur par défaut : réplication temps réel et synchronisation automatique (pas de planification) Utiliser l'attribut `syncat` pour planifier la synchronisation des répertoires répliqués sur le nœud secondaire à des dates/heures données. Pour plus de détails, voir section 13.6.4.10. Le module doit être démarré pour activer cette fonctionnalité. Une fois synchronisé, le module se bloque dans l'état `WAIT (NotReady)` jusqu'à la prochaine synchronisation. La planification est basée sur le gestionnaire de tâches du système d'exploitation : • en Windows, la tâche est définie comme tâche système • en Linux, la tâche est insérée dans la `crontab` de l'utilisateur `safekit` Vous devez simplement configurer `syncat` avec la syntaxe du gestionnaire de tâche du système. Par exemple, pour une synchronisation quotidienne, après minuit : • en Windows `syncat="/SC DAILY /ST 00:01:00"` • en Linux `syncat="01 0 * * *"`

	 Voir la documentation de `crontab` en Unix et de `schtasks.exe` en Windows, pour une description complète de la syntaxe.  Si la configuration ou la planification ne fonctionnent pas correctement, vérifier d'abord les erreurs de syntaxe ou d'utilisation du gestionnaire de tâches du système.
`[<flow name="network" > <server addr="IP_1" /> <server addr="IP_2" /> </flow>]`	**Ancienne** configuration préservée pour la compatibilité ascendante. Quand cette section n'est pas définie, le flux de réplication passe par le heartbeat défini avec ident="`flow`" s'il y en a un, sinon par la voie du 1er heartbeat (pour la description des heartbeats, voir section 13.3). Si vous utilisez cette configuration, il faut assurer la cohérence avec la définition d'un heartbeat avec `ident=flow`" car des règles de failover par défaut sont définies (décrites en section 13.18.4).  Le sous-arbre `<flow>` peut être modifié dynamiquement, pour changer le flux de réplication par exemple. L'attribut name de `<flow>` nommé le réseau utilisé pour le flux de réplication. `network` doit être le nom d'un réseau défini dans la configuration du cluster global (voir section 12). Le tag `<server>` était utilisé dans l'ancienne syntaxe de configuration (avant SafeKit 7.2). Il est supporté pour assurer la compatibilité ascendante, mais ne doit pas être utilisé pour la configuration de nouveaux modules.  Vous ne devez pas utiliser dans le même `userconfig.xml`, la syntaxe de SafeKit 7.1 et celle introduite depuis SafeKit 7.2.
`<replicated`	Définition des répertoires répliqués Mettre autant de sections que de répertoires à répliquer
`dir="`*`abs_path`*`"`	Path absolu du répertoire à répliquer.
`[mode= "read_only"]`	Accès en read-only sur la machine secondaire pour éviter la corruption.
`<notreplicated path="`*`relative`*`" />`	Path relatif d'un fichier ou d'un sous répertoire d'un répertoire répliqué. Le fichier (ou sous répertoire) est non répliqué. Autant de lignes que de fichiers ou sous répertoire à non répliquer.
`<notreplicated regexpath="`*`expression régulière`*`" />`	Expression régulière sur le nom des entrées sous le répertoire répliqué : • Réplication du contenu du répertoire **excepté** les entrées qui correspondent à l'expression régulière

	Par exemple, pour ne pas répliquer les entrées dont l'extension est `.tmp` ou `.bak` dans le répertoire `/safedir` ou ses sous-répertoires : `<replicated dir="/safedir">` `<notreplicated regexpath=".*\.tmp$" />` `<notreplicated regexpath=".*\.bak$" />` `</replicated>` Notez que `/safedir/conf/config.tmp.swap` est répliqué. • Réplication dans le répertoire **uniquement** des entrées qui correspondent par l'expressions régulière après le **!** Par exemple, pour ne répliquer que les entrées dont l'extension est `.mdf` ou `.ldf` dans le répertoire `/safedir` ou ses sous-répertoires : `<replicated dir="/safedir">` `<notreplicated regexpath="!.*\.mdf$" />` `<notreplicated regexpath="!.*\.ldf$" />` `</replicated>` Le renommage entre des fichiers non répliqués et répliqués n'est pas supporté. Le moteur d'évaluation des expressions régulière est POSIX Extended regex (voir la documentation POSIX) : • en Windows, mode insensible à la casse • en Linux, mode sensible à la casse Comme les expressions régulières sont définies dans un fichier XML, les caractères spéciaux interprétés par XML comme '<' ou '>' ne peuvent pas être utilisés dans les expressions régulières.
`<tocheck path="relative" />`	Path relatif d'un fichier ou d'un sous répertoire dans un répertoire répliqué. Vérifier sa présence avant de démarrer. Evite un démarrage sur un répertoire vide. Mettre autant de lignes que nécessaire.

13.6.4 <rfs>Description

13.6.4.1 <rfs> prérequis

Voir les prérequis décrits en section 2.2.4.

En Windows, activez le journal USN sur le lecteur qui contient les répertoires répliqués afin d'activer la réintégration par zones, après redémarrage du serveur, à condition que le module ait été arrêté proprement.

13.6.4.2 <rfs> Linux

Sur Linux, l'interception des données répliquées est basée sur un montage NFS local. Et le flux de réplication entre les serveurs est basé sur le protocole NFS v3 / TCP.

Le montage NFS des répertoires répliqués à partir de clients Linux externe n'est pas supporté. En revanche, le montage d'autres répertoires peut être réalisé avec les commandes standards.

Procédure pour répliquer un point de montage

Quand un répertoire répliqué est un point de montage, la configuration du module échoue avec l'erreur suivante :

```
Erreur: périphérique ou ressource occupée
```

Dans la suite, nous prenons l'exemple du module PostgreSQL qui définit en tant que répertoires répliqués `/var/lib/pgsql/var` et `/var/lib/pgsql/data`. Le fichier `userconfig.xml` du module contient :

```
<rfs … >
  <replicated dir="/var/lib/pgsql/var" mode="read_only" />
  <replicated dir="/var/lib/pgsql/data" mode="read_only" />
</rfs>
```

Ces répertoires sont des points de montage comme le montre le résultat de la commande `df -H`. La commande retourne par exemple :

```
/dev/mapper/vg01-lv_pgs_var … /var/lib/pgsql/var
/dev/mapper/vg02-lv_pgs_data … /var/lib/pgsql/data
```

Vous devez appliquer la procédure suivante pour configurer le module avec la réplication de ces répertoires.

C'est la même procédure pour tous les points de montage qui doivent être répliqués.

1. démonter les systèmes de fichiers en exécutant :

   ```
   umount /var/lib/pgsql/var
   ```

   ```
   umount /var/lib/pgsql/data
   ```

2. configurer le module en exécutant :

   ```
   /opt/safekit/safekit config -m postgresql
   ```

   La configuration se termine avec succès.

3. vérifier l'existence des liens symboliques créés lors de la configuration en exécutant `ls -l /var/lib`. La commande retourne :

   ```
   lrwxrwxrwx 1 root root var -> var_For_SafeKit_Replication
   lrwxrwxrwx 1 root root data -> data_For_SafeKit_Replication
   ```

4. éditer `/etc/fstab` et modifier les 2 lignes :

   ```
   /dev/mapper/vg01-lv_pgs_var /var/lib/pgsql/var ext4…
   /dev/mapper/vg02-lv_pgs_data /var/lib/pgsql/data ext4…
   ```

   par

   ```
   /dev/mapper/vg01-lv_pgs_var /var/lib/pgsql/var_For_SafeKit_Replication ext4…
   /dev/mapper/vg02-lv_pgs_data /var/lib/pgsql/data_For_SafeKit_Replication
   ext4..
   ```

5. monter les systèmes de fichiers en exécutant :

   ```
   mount /var/lib/pgsql/var_For_SafeKit_Replication
   ```

   ```
   mount /var/lib/pgsql/data_For_SafeKit_Replication
   ```

Appliquez cette procédure sur les deux nœuds si les répertoires répliqués sont des points de montage sur les deux nœuds. Une fois appliquée, vous pouvez utiliser le module comme d'habitude : par exemple safekit start stop etc ...

Pour empêcher le démarrage du module quand le répertoire est non monté et vide, vous pouvez insérer dans `userconfig.xml` la vérification de la présence d'un fichier dans le répertoire répliqué. Exemple pour `/var/lib/pgsql/var` (faire de même pour `/var/lib/pgsql/data` en testant un fichier toujours présent dans ce répertoire) :

```
<replicated dir="/var/lib/pgsql/var" mode="read_only">
    <tocheck path="postgresql.conf" />
</replicated>
```

A la déconfiguration du module (ou désinstallation du package SafeKit), vous devez appliquer la procédure inverse pour restaurer l'état initial :

1. démonter les systèmes de fichiers en exécutant :

   `umount /var/lib/pgsql/var_For_SafeKit_Replication`

   `umount /var/lib/pgsql/data_For_SafeKit_Replication`

2. déconfigurer le module en exécutant /opt/safekit/safekit deconfig -m postgresql
3. éditer /etc/fstab pour y restaurer son état initial
4. monter les systèmes de fichiers en exécutant :

   `mount /var/lib/pgsql/var`

   `mount /var/lib/pgsql/data`

13.6.4.3 <rfs> Windows

Sur Windows, l'interception des données est basée sur un filtre file system. Et le flux de réplication entre les serveurs est basé sur le protocole NFS v3 / TCP.

Certains anti-virus peuvent empêcher le fonctionnement correct de la réplication.

Sur Windows, il est possible de monter à distance un répertoire répliqué. Si vous voulez pouvoir monter avec le nom et non l'adresse IP virtuelle, vous devez positionner les valeurs suivantes dans les bases de registre des deux serveurs SafeKit :

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
"DisableLoopbackCheck"=dword:00000001
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\lanmanserver\parameters] "DisableStrictNameChecking"=dword:00000001
```

En Windows, pour activer la réintégration par zones après le redémarrage du serveur, lorsque le module a été correctement arrêté, le composant <rfs> utilise le journal NTFS USN pour vérifier que les informations enregistrées sur les zones sont toujours valables après le redémarrage. Lorsque le contrôle réussit, la réintégration par zones peut être appliquée sur le fichier ; sinon, le fichier doit être recopié dans sa totalité.

Par défaut, seul le lecteur système a un journal USN actif. Si les répertoires répliqués sont situés sur un lecteur différent du lecteur système, vous devez créer le journal (avec commande `fsutil usn`).

Voir SK-0066 pour un exemple.

13.6.4.4 <rfs> Réplication et reprise sur panne

Avec la réplication de fichiers, l'architecture miroir est particulièrement adaptée à la haute disponibilité des applications base de données avec des données critiques à

protéger contre les pannes. En effet, les données du serveur secondaire sont fortement synchronisées avec celles du serveur primaire. Le serveur est dit à jour et seul un serveur à jour peut démarrer en primaire ou effectuer une reprise sur panne

Si la disponibilité de l'application est plus critique que la synchronisation des données, la politique par défaut peut être relâchée pour autoriser un serveur non à jour à devenir primaire mais uniquement si la date de la dernière synchronisation est inférieure à un délai configurable. Cela est configuré avec l'attribut `syncdelta` du tag <rfs> dont la valeur est exprimée en minutes :

- `syncdelta` <= 1

  L'attribut est ignoré et la politique par défaut de démarrage en primaire et de reprise sur panne est appliquée. La valeur par défaut 0.

- `syncdelta` > 1

  Si le serveur à jour ne répond pas, le serveur non à jour peut devenir primaire mais uniquement si le temps écoulé depuis la dernière synchronisation est inférieur à la valeur de `syncdelta` (en minutes).

Cette fonctionnalité est implémentée à l'aide de :

- la ressource `rfs.synced`

  Quand `syncdelta` est > 1, la gestion de la ressource `rfs.synced` est activée. Cette ressource est dans l'état `UP` si les données répliquées sont cohérentes et le temps écoulé depuis la dernière synchronisation est inférieur à la valeur de `syncdelta`.

- Le checker `syncedcheck`

  Quand `syncdelta` est > 1, ce checker est activé. Il affecte la valeur de la ressource `rfs.synced`.

- La règle de failover `rfs_forceuptodate`

  Quand `syncdelta` est > 1, la règle de failover suivante est valide :

  rfs_forceuptodate: if (`heartbeat`. == down && cluster() == down && rfs.synced == up && rfs.uptodate == down) then rfs.uptodate=up;*

  Cette règle provoque le démarrage en primaire du serveur lorsque le serveur à jour ne répond pas, et à condition que ce serveur soit isolé et considéré synchronisé en fonction de la valeur de `syncdelta`.

13.6.4.5 <rfs> Vérification de la réplication

Vous pouvez vérifier que les fichiers sont identiques sur le primaire et le secondaire avec la commande suivante à passer sur la machine `SECOND` : `safekit rfsverify -m AM`. Exécuter `safekit rfsverify -m AM > log` pour rediriger la sortie de la commande dans un fichier nommé `log`.

La sortie de la commande est un journal similaire à celui de la réintégration dans lequel sont indiqués les fichiers à recopier (donc différents).

Quand sur la primaire, il y a de l'activité sur les répertoires répliqués, il se peut qu'une anomalie soit détectée alors qu'il n'y a pas de différence entre les fichiers. Cela se produit dans les cas suivants :

- sur Windows à cause des modifications faites sur disque avant d'être répliquées,

- avec `async="second"` (défaut) car les lectures peuvent dépasser les écritures.

Pour vérifier s'il y a vraiment une incohérence, vous devez relancer la commande sur le serveur secondaire en s'assurant qu'il n'y plus d'activité sur le serveur primaire.

Sur Windows, certains fichiers modifiés avec l'option `SetvalidData` sont systématiquement détectés différents car ils sont étendus sans reset des données : le contenu en lecture des zones étendues est le contenu aléatoire du disque au moment de la lecture.

Il est fortement recommandé d'exécuter cette commande uniquement lorsqu'il n'y a pas d'accès aux répertoires répliqués sur le primaire.

13.6.4.6 <rfs> Fichiers modifiés depuis la dernière synchronisation

Avant de démarrer le serveur secondaire, il peut être utile d'évaluer le nombre de fichiers et la quantité de données qui ont été modifiés sur le serveur primaire depuis l'arrêt du serveur secondaire. Cette fonctionnalité est fournie en exécutant la commande suivante sur le serveur `ALONE` : `safekit rfsdiff -m` *`AM`*. Exécuter `safekit rfsdiff -m` *`AM`* `> log` pour rediriger la sortie de la commande dans un fichier nommé `log`.

Cette commande exécute des vérifications en ligne du contenu des fichiers réguliers du module *`AM`*. Elle analyse l'arborescence répliquée entièrement et affiche le nombre de fichiers qui ont été modifiés ainsi que la taille qui doit être recopiée. Elle affiche également une estimation du temps total de réintégration. Ceci n'est qu'une évaluation car seuls les fichiers réguliers sont analysés et d'autres modifications peuvent se produire jusqu'à ce que la synchronisation soit exécutée par le serveur secondaire.

Cette commande doit être utilisée avec précaution sur un serveur en production car elle entraîne une surcharge sur le serveur (pour la lecture, avec verrouillage, de l'arborescence et des fichiers). En Windows, le renommage des fichiers peut échouer pendant cette évaluation.

Il est fortement recommandé d'exécuter cette commande uniquement lorsqu'il n'y a pas d'accès aux répertoires répliqués.

13.6.4.7 <rfs> Bande passante de réplication et de réintégration

Le composant de réplication collecte sur le serveur `PRIM` la bande passante utilisée par les opérations d'écritures de réplication et de réintégration.

Deux ressources (`rfs_bandwidth.replication` et `rfs_bandwidth.reintegration`), exprimées en kilo octet par seconde (KB/s), reflètent la bande passante moyenne utilisée respectivement par la réplication et la réintégration durant les 3 dernières secondes.

Si la charge de réplication est très active en écriture, une saturation du lien réseau peut se produire lors de la phase de réintégration, entraînant un ralentissement significatif de l'application. Dans ce cas, l'attribut `<rfs> reiallowedbw` peut être utilisé pour limiter la bande passante utilisée par la phase de réintégration (voir section 13.6.3). Il faut cependant considérer que la limitation de la bande passante de réintégration allongera la durée de la phase de réintégration.

Il y a aussi 2 nouvelles ressources qui reflètent la bande passante réseau (en KOctets/sec), utilisée entre les processus `nfsbox`, qui s'exécutent sur chaque nœud pour implémenter la réplication et la réintégration :

- `rfs.netout_bandwidth` est la bande passante utilisée en sortie
- `rfs.netin_bandwidth` est la bande passante utilisée en entrée

Vous pouvez observer la valeur de `rfs.netout_bandwidth` sur le primaire ou de `rfs.netin_bandwidth` sur le secondaire pour connaître le taux de modification au moment de l'observation (écriture, création, suppression, ...). L'historique des valeurs de la ressource donne un aperçu de son évolution dans le temps.

La valeur de la bande passante dépend de l'activité applicative, système et réseau. Sa mesure n'est disponible qu'à titre d'information.

13.6.4.8 <rfs> Synchronisation par date

Depuis SafeKit 7.2, SafeKit offre la nouvelle commande `safekit secondforce -d date -m AM` qui force le module *AM* à démarrer comme secondaire après avoir copié uniquement les fichiers modifiés après la date spécifiée.

Cette commande doit être utilisée avec précautions car la synchronisation ne copiera pas les fichiers modifiés avant la date spécifiée. Il incombe à l'administrateur de s'assurer que ces fichiers sont cohérents et à jour.

La date est dans le format YYYY-MM-DD[Z] ou "YYYY-MM-DD hh:mm:ss[Z]" ou YYYY-MM-DDThh:mm:ss[Z], où:

- YYYY-MM-DD indique l'année, le mois et le jour
- hh:mm:ss indique l'heure, les minutes et secondes
- Z indique que la date est exprimée dans le fuseau horaire UTC ; s'il n'est pas spécifié, la date est exprimée dans le fuseau horaire local

Par exemple :

- `safekit secondforce -d 2016-03-01 -m AM` copie uniquement les fichiers modifies après le 1er Mars 2016
- `safekit secondforce -d "2016-03-01 12:00:00" -m AM` copie uniquement les fichiers modifies après le 1er Mars 2016 à 12h, heure locale
- `safekit secondforce -d 2016-03-01T12:00:00Z -m AM` copie uniquement les fichiers modifies après le 1er Mars 2016 à 12h, dans le fuseau horaire UTC

Cette commande peut être utile dans le cas suivant :

- le module est arrêté sur le serveur primaire et une sauvegarde des données répliquées est effectuée (sur un lecteur amovible par exemple)
- le module est arrêté sur le serveur secondaire et les données répliquées sont restaurées à partir de la sauvegarde. Il peut s'agir du premier démarrage ou de la réparation du serveur secondaire.
- le module est démarré sur le serveur primaire qui devient `ALONE`
- le module est démarré sur le secondaire avec la commande `safekit secondforce -d date -m AM` où la date est la date de sauvegarde

Dans ce cas, seuls les fichiers modifiés depuis la date de la sauvegarde seront copiés (dans leur totalité), au lieu de la copie complète de tous les fichiers.

En Windows, la date de modification du fichier sur le serveur secondaire est affectée lorsque le fichier est copié par le processus de réintégration. Par conséquent, `safekit secondforce -d date -m AM`, dont la date est antérieure à la dernière réintégration sur ce serveur, n'a aucun intérêt.

13.6.4.9 <rfs> Synchronisation externe

Lors de la première synchronisation, tous les fichiers répliqués sont copiés dans leur totalité du nœud principal vers le nœud secondaire. Lors des synchronisations suivantes, nécessaires lors du redémarrage du nœud secondaire, seules les zones modifiées des fichiers sont recopiées. Lorsque les répertoires répliqués sont volumineux, la première synchronisation peut prendre beaucoup de temps en particulier si le réseau est lent. C'est pourquoi, depuis SafeKit> 7.3.0.11, SafeKit fournit une nouvelle fonctionnalité pour synchroniser un grand volume de données qui doit être utilisée conjointement avec un outil de sauvegarde.

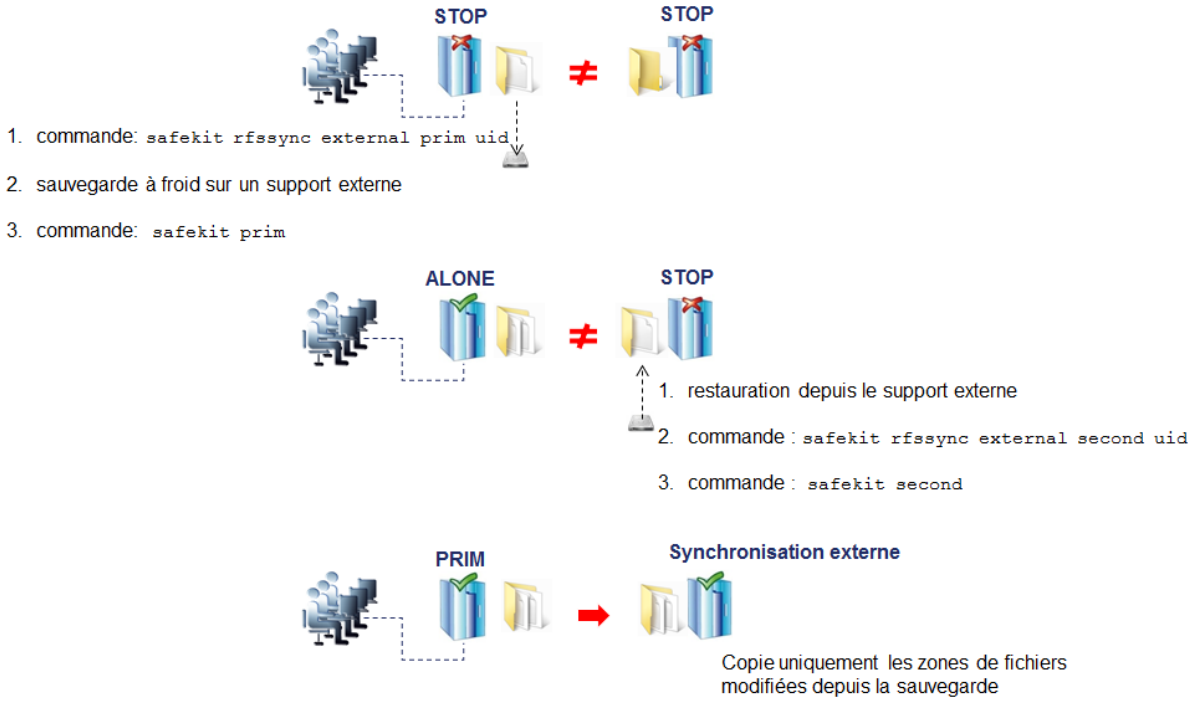
Sur le nœud principal, il suffit d'effectuer une sauvegarde des répertoires répliqués et de passer la politique synchronisation au mode externe. La sauvegarde est transportée (en utilisant un lecteur externe par exemple) et restaurée sur le nœud secondaire, qui est aussi configuré pour effectuer une synchronisation externe. Lorsque le module est démarré sur le nœud secondaire, il recopie uniquement les zones de fichiers modifiées sur le nœud principal depuis la sauvegarde.

La synchronisation externe repose sur une nouvelle commande `safekit rfssync` qui doit être appliquée sur les deux nœuds afin de positionner le mode de synchronisation à `external`. Cette commande prend comme arguments :

- le rôle du nœud (`prim | second`)
- un identificateur unique (`uid`)

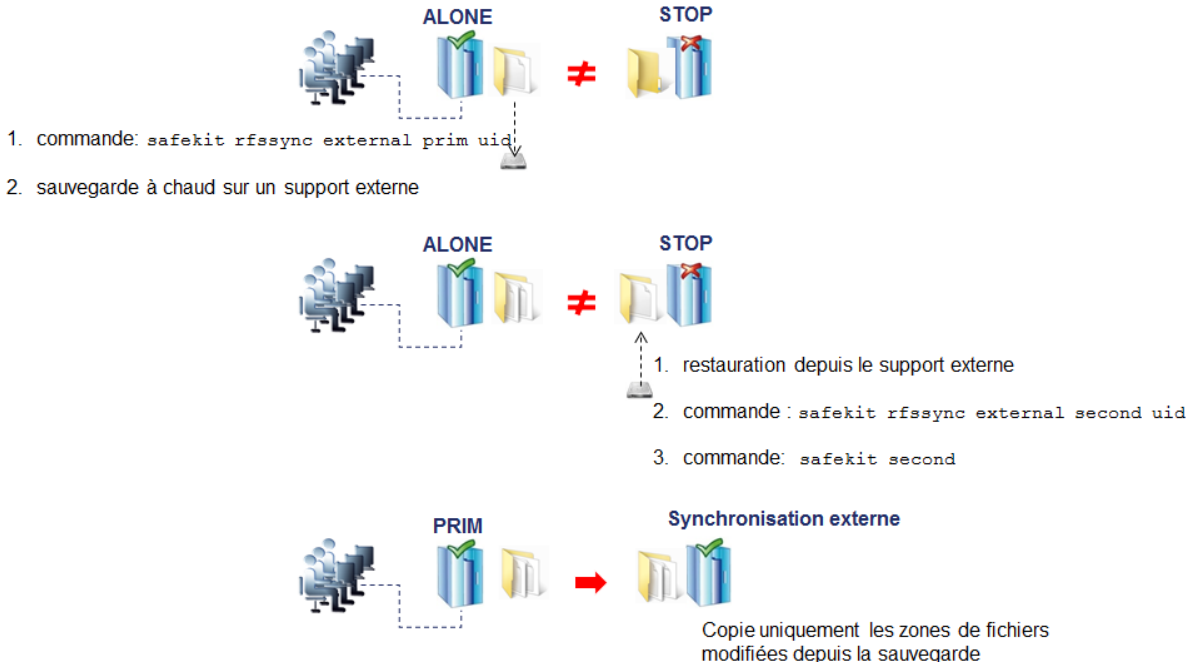
Procédure de synchronisation externe

La procédure de synchronisation externe, décrite ci-dessous, est la procédure à appliquer dans le cas d'une sauvegarde à froid des répertoires répliqués. Dans ce cas, l'application doit être arrêtée et toute modification des répertoires répliqués est interdite jusqu'au démarrage du module, et de l'application, en `ALONE(Ready)`. L'ordre des opérations doit être strictement respecté.



La procédure de synchronisation externe, décrite ci-dessous, est la procédure à appliquer dans le cas d'une sauvegarde à chaud des répertoires répliqués. Dans ce cas, le module

est `ALONE (Ready)` ; l'application est démarrée et les modifications du contenu des répertoires répliqués sont autorisées. L'ordre des opérations doit être strictement respecté.



Commande safekit rfssync

`safekit rfssync external prim` *`uid`* `[-m` *`AM`*`]`	Positionne la politique de synchronisation à `external`. Elle est identifiée par la valeur de *`uid`* (max 24 char). Le nœud est le primaire, source pour la synchronisation des données.
`safekit rfssync external second` *`uid`* `[-m` *`AM`*`]`	Positionne la politique de synchronisation à `external`. Elle est identifiée par la valeur de `uid` (max 24 char). Le nœud est le secondaire, destination de la synchronisation des données.
`safekit rfssync -d prim` *`uid`* `[-m` *`AM`*`]` `safekit rfssync -d second` *`uid`* `[-m` *`AM`*`]`	Désactive le contrôle des modifications des répertoires répliqués entre le moment de la sauvegarde à froid/la restauration et le démarrage du module. Cette option doit être utilisée avec précautions puisque la synchronisation externe peut dans ce cas ne pas détecter toutes les modifications à recopier.
`safekit rfssync full [-m` *`AM`*`]`	Positionne la politique de synchronisation à `full`. Celle-ci entraîne la recopie de tous les fichiers dans leur totalité à la prochaine synchronisation.
`safekit rfssync`	Affiche la politique de synchronisation courante.

Internes

La politique de synchronisation est représentée par des ressources du module : `usersetting.rfssyncmode`, `usersetting.rfssyncrole`, `usersetting.rfssyncuid` et `rfs.rfssync` :

- `usersetting.rfssyncmode="default"`

  `(usersetting.rfssyncrole="default", usersetting.rfssyncuid="default")`

  Ces valeurs sont associées à la politique de synchronisation standard, celle appliquée par défaut. Elle consiste à ne copier que les zones modifiées des fichiers. Quand cette stratégie ne peut être appliquée, les fichiers modifiés sont recopiés dans leur totalité.

- `usersetting.rfssyncmode="full"`

  (`usersetting.rfssyncrole="default"`, `usersetting.rfssyncuid="default"`)

  Ces valeurs sont associées à la politique de synchronisation `full`. Elle est appliquée :

  - au premier démarrage du module après sa première configuration
  - sur commandes `safekit` (`safekit second fullsync` ; `safekit rfssync full ; safekit primforce ; safekit config ; safekit deconfig`)
  - sur changement d'appariement du module

  La politique de synchronisation `full` entraîne la recopie de tous les fichiers dans leur totalité à la prochaine synchronisation.

- `usersetting.rfssyncmode="external"`, `usersetting.rfssyncrole="prim" | "second"` and `usersetting.rfssyncuid="uid"`

  Ces valeurs sont associées à la politique de synchronisation `external` affectée avec les commandes `safekit rfssync external prim uid` ou `safekit rfssync external second uid`. La prochaine synchronisation appliquera la politique de synchronisation externe.

- `rfs.rfssync="up" | "down"`

  Cette ressource vaut `up` uniquement lorsque la politique de synchronisation, définie par les ressources précédentes, peut être appliquée.

Quand la politique de synchronisation n'est pas celle par défaut, celle-ci repasse automatiquement dans le mode par défaut une fois la synchronisation appliquée avec succès.

Dans certains cas, la synchronisation externe ne peut être appliquée et le nœud secondaire s'arrête avec une erreur indiquée dans le journal du module. Dans cette situation, il faut soit :

- compléter la procédure de synchronisation externe si celle-ci n'a pas été effectuée dans sa totalité sur les 2 nœuds
- réappliquer complètement la procédure de synchronisation sur les 2 nœuds
- appliquer la politique de synchronisation `full` (commande `safekit rfssync full`)
- appliquer la synchronisation par date, en utilisant la date de la sauvegarde (voir section 13.6.4.8). Contrairement à la synchronisation externe, la synchronisation par date va copier dans leur totalité (et non par zones) les fichiers modifiés sur le nœud primaire.

13.6.4.10 <rfs> Synchronisation planifiée

Par défaut, SafeKit offre la réplication de fichiers en temps réel et une synchronisation automatique. Si la charge est importante sur le nœud primaire ou si le réseau a une forte latence, il peut être préférable d'accepter que le nœud secondaire ne soit pas fortement synchronisé avec le nœud primaire. Pour cela, vous pouvez utiliser l'attribut `syncat` pour planifier une synchronisation régulière des répertoires répliqués sur le nœud secondaire. Le module doit être démarré pour activer cette fonctionnalité. Une fois synchronisé, le module se bloque dans l'état `WAIT (NotReady)` jusqu'à la prochaine synchronisation planifiée. Cette fonctionnalité est implémentée avec :

- la ressource `rfs.syncat` affectée à `up` aux dates planifiées et affectée à `down` une fois le nœud secondaire synchronisé
- la règle de failover `rfs_syncat_wait` qui bloque le nœud secondaire dans l'état `WAIT (NotReady)` jusqu'à ce que la ressource `rfs.syncat` soit `up`

Si vous souhaitez forcer la synchronisation en dehors des dates planifiées, il faut exécuter la commande `safekit set -r rfs.syncat -v up -m` *AM* quand le module est dans l'état `WAIT (NotReady)`.

La configuration de `syncat` se fait simplement en utilisant la syntaxe du gestionnaire de tâches du système d'exploitation : `crontab` en Linux et `schtasks.exe` en Windows (voir section 13.6.3).

13.7 Activer les scripts du module - <user>, <var>

Cette section décrit uniquement les options de configuration du tag `<user>`. Pour une description complète des scripts, voir la section 14.

13.7.1 <user> Exemple

```
<user logging="userlog" >
      <var name="name1" value="value1" />
</user>
```

Pour un exemple de l'utilisation de `<var>`, voir la section 15.3. Voir aussi l'exemple complet d'un module miroir en section 15.1, et d'un module ferme en section 15.2. Elles présentent la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.7.2 <user> Syntaxe

```
<user
   [nicestoptimeout="300"]
   [forcestoptimeout="300"]
   [logging="userlog"|"none"]
   [userlogsize="2048"]
 >
      [<var name="name1" value="value1" />]
      …
</user>
```

Le tag `<user>` et son sous-arbre peuvent être entièrement modifiés dynamiquement.

13.7.3 <user>, <var> Attributs

<user	
[nicestoptimeout="300"]	Timeout en secondes pour exécuter les scripts stop_xx. Valeur par défaut : 300 secondes
[forcestoptimeout="300"]	Timeout en secondes pour exécuter les scripts stop_xx - force Valeur par défaut : 300 secondes
[logging="userlog"|="none"]	• logging="userlog" : les messages stdout et stderr de l'application démarrée dans les scripts redirigés dans le journal des scripts dans le fichier SAFEVAR/modules/*AM*/userlog_<year>_<month>_<day>T<time>_<script name>.ulog où *AM* est le nom du module (SAFEVAR=C:\safekit\var sur Windows et /var/safekit sur LINUX). • logging="none" : les messages stdout et stderr de l'application démarrée dans les scripts ne sont pas logués Valeur par défaut : userlog
[userlogsize="2048"]	Taille limite en KO du journal des scripts. Au démarrage du module, le fichier est reseté si sa taille a dépassé la limite. Valeur par défaut : 2048 KO
<var name="ENV_VARIABLE_1" value="VALUE_1" />	Variable d'environnement et sa valeur exportée avant l'exécution des scripts. Mettre autant de lignes que de variables.

13.8 Hostname virtuel - <vhost>, <virtualhostname>

13.8.1 <vhost> Exemple

```
<vhost>
  <virtualhostname name="vhostname" envfile="vhostenv" />
</vhost>
```

Voir l'exemple en section 15.13. Elle présente la configuration via la console web ainsi que le userconfig.xml correspondant.

13.8.2 <vhost> Syntaxe

```
<vhost>
  <virtualhostname
     name="virtual_hostname"
     envfile="path_of_a_file"
    [when="prim"|"second"|"both"]
```

```
  />
</vhost>
```

Le tag `<vhost>` et son sous-arbre **ne peuvent pas** être modifiés dynamiquement.

13.8.3 <vhost>, <virtualhostname> Attributs

`<vhost>`	
`<virtualhostname`	
`name="`*`virtual_hostname`*`"`	Définition du nom virtuel.
`envfile="`*`path_of_envfile`*`"`	Chemin d'un fichier d'environnement généré automatiquement par SafeKit à la configuration. Si le chemin du fichier est relatif, le fichier est généré dans les fichiers d'environnement du module, i.e. : `SAFEUSERBIN` Ce fichier est utilisé dans les scripts pour positionner le hostname virtuel. Voir le module `vhost.safe` livré avec le package Linux et Windows.
`[when="prim"|"second"|"both"]`	Définis quand le hostname virtuel doit être rendu. Par défaut, `prim` signifie quand le module est primaire (`PRIM` ou `ALONE`).
`/>`	
`</vhost>`	

13.8.4 <vhost> Description

Certaines applications ont besoin de voir le même hostname quel que soit le serveur d'exécution (typiquement, elles stockent le hostname dans un fichier répliqué). Le hostname virtuel peut être présenté à ces applications alors que les autres applications voient le hostname physique des serveurs.

- Sur Linux

  La mise en œuvre est basée sur la variable d'environnement LD_PRELOAD : les fonctions `gethostname` et `uname` sont surchargées.

- Sur Windows
- La mise en œuvre est basée sur la variable d'environnement CLUSTER_NETWORK_NAME_ : les fonctions de l'API name query (GetComputerName, GetComputerNameEx, gethostname) sont surchargées.

  Pour utiliser vhost avec un service, utiliser les commandes `vhostservice <service> [<file>]` avant/après le démarrage/arrêt du service dans les scripts du module.

13.9 Surveillance de processus ou services - <errd>, <proc>

La section `<errd>` nécessite d'avoir défini la section `<user/>`.

13.9.1 <errd> Exemple

Voir aussi un exemple complet en section 15.4. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.9.1.1 Surveillance de processus

- Linux et Windows

  `myproc` est le nom de la commande associée au processus à surveiller.

```
<errd>
  <proc name="myproc" atleast="1" action="restart" class="prim" />
</errd>
```

- Linux uniquement (pour SafeKit > 7.2.0.29)

  `oracle_.*` est une expression régulière sur le nom de la commande associée au processus à surveiller.

```
<errd>
  <proc name="oracle" nameregex="oracle_.*" atleast="1" action="restart"
class="prim"/>
</errd>
```

13.9.1.2 Surveillance de service

`myservice` est le nom du service à surveiller . En Windows (pour safekit > 7.3), il s'agit du nom du service Windows. En Linux, il s'agit du nom du service systemd (pour safekit > 7.4.0.19).

```
<errd>
  <proc name="myservice" service="yes" action="restart" class="prim" />
</errd>
```

13.9.2 <errd> Syntaxe

```
<errd
  [polltimer="10"]
>
  <proc name="command name and/or resource name for the monitored process or
service"
        [service="no|yes"]
        [nameregex=="regular expression on the command name"]
        [argregex="regular expression on process arguments, including command
name"]
        atleast="1"
        action="stopstart"|"restart"|"stop"|"executable_name"
        class="prim"|"both"|"pre"|"second"|"sec"|"othername"]
        [start_after="nb polling cycles"]
        [atmax="-1"]
  />
  …
</errd>
```

Le tag <errd> et son sous-arbre peuvent être entièrement modifiés dynamiquement.

13.9.3 <errd>, <proc> Attributs

<errd	
polltimer="30"	Temps de polling, en secondes, entre 2 surveillances des processus. Valeur par défaut : 30 secondes
<proc	Définition du processus à surveiller. Autant de lignes que de processus. Une ressource est associée à chaque <proc> et est nommée proc.<valeur de l'attribut name> (par exemple proc.process_name). La ressource vaut up lorsque la condition de surveillance est vraie ; vaut down sinon.
name="*command_name*"	*command_name* est le nom de la commande associée au processus à surveiller. C'est aussi le nom de la ressource associée au processus surveillé. Au maximum 15 caractères pour Linux (le nom de la commande peut être tronqué) ; 63 pour Windows. Exemple : • sur LINUX, name="vi" • sur Windows name="notepad.exe". En Windows, le nom est automatiquement converti en minuscule. Pour retrouver le nom des commandes associées aux processus, voir les commandes décrites en section 13.9.4
Ou name="*service_name*" service="yes"	*service_name* est le nom du service à surveiller. C'est aussi le nom de la ressource associée au service surveillé. Au maximum, 63 caractères. Exemple : • name="W32Time" service="yes" surveille le service de Temps Windows. • name="ntpd" service="yes" surveille le service de Temps Linux (systemd ntpd.service) . L'attribut service est facultatif et sa valeur par défaut est : no.

Ou `name="resource_name"` `nameregex="regular expression on the command name"`	**Linux uniquement** `nameregex` est une expression régulière sur le nom de lacommande pour sélectionner le processus à surveiller. `name` est le nom de la ressource associée au processus surveillé. Comme les expressions régulières sont définies dans le fichier XML `userconfig.xml`, certains caractères ne peuvent pas être utilisés '`<`' ou '`>`'. Exemple : • `nameregex = "oracle _. *" name = "oracle"` surveille les processus oracle dont le nom de la commande respecte l'expression régulière La ressource associée est `proc.oracle` L'attribut `nameregex` est facultatif.
`class=` `"prim"|` `"both"|` `"pre"|` `"second"|` `"sec"|` `"othername"`	Le processus appartient à une classe. La surveillance est activée/désactivée avec la commande `safekit errd enable|disable classname -m AM`. • `class="prim"|"both"|"pre"|"second"|"sec"` L'activation/désactivation de ces classes est automatique et faite dans le composant `<user/>` après/avant `start_prim/stop_prim`, `start_both/stop_both`, `start_second/stop_second`, `start_sec/stop_sec`. Pour une description des scripts, voir section 14. • `class="othername"` Avec un autre nom de classe, l'activation/désactivation doit être faite explicitement après/avant le démarrage/arrêt des processus de la classe.
`[argregex="regular expression on process arguments"]`	Expression régulière sur la liste des arguments du processus incluant le nom de l'exécutable. Le moteur d'évaluation des expressions régulière est POSIX Extended regex (voir la documentation POSIX) : • en Windows, mode insensible à la casse • en Linux, mode sensible à la casse Comme les expressions régulières sont définies dans le fichier XML `userconfig.xml`, certains caractères ne peuvent pas être utilisés '`<`' ou '`>`'.

	Pour retrouver la liste des arguments d'un processus, utiliser les commandes décrites en section 13.9.4. • Exemple en Linux avec l'éditeur `vi` sur le fichier `myfile` `<proc name="vi" argregex=".*myfile.*" …` `<proc name="vi" argregex="/myrep/myfile.*" …` `<proc name="vi" argregex="/myrep/myfile" …` • Exemple en Windows avec l'éditeur `notepad` sur le fichier `myfile` `<proc name="notepad.exe" argregex=".*myfile.*" …` `<proc name="notepad.exe" argregex="c:\\myrep\\myfile.*" …` `<proc name="notepad.exe" argregex="c:\\myrep\\myfile" …`
`atleast="1"`	Nombre minimum de processus qui doivent s'exécuter. Si le minimum est atteint, SafeKit déclenche l'action. • `name="oracle" argregex=".*db1.*" atleast="1"` signifie qu'une action est déclenchée si 0 processus s'exécute sur l'instance `oracle/db1`. • `atleast="-1"` l'attribut n'est pas pris en compte. Valeur par défaut : `1`
`action= "restart"| "stopstart"| "stop"| "noaction"| "executable_name"`	Action (ou handler) à exécuter sur le module • `action="restart"` provoque le `restart` • `action="stopstart"` provoque le `stopstart` et éventuellement un basculement • `action="stop"` provoque le `stop` et éventuellement un basculement Les actions `restart/stopstart` incrémentent le compteur `maxloop` pour vérifier que l'on n'est pas sur une faute reproductible. Pour la description de `maxloop`, voir section 13.2. • `action="noaction"` ne provoque pas d'action mais seulement un message dans le journal • `action="executable_name"` Pour un handler, soit mettre le chemin absolu du handler, soit mettre le chemin relatif au répertoire bin du module ("`SAFE/modules/AM`/bin/"). Nous conseillons un chemin relatif avec un handler défini dans le module.

	Avec un handler spécial, une classe avec un nom propre doit être définie. Pour un handler spécial en Linux, insérer à la fin du script, `exit 0` Pour un handler spécial en Windows, mettre à la fin `%SAFEBIN%\exitcode 0`. Sinon, c'est un échec et SafeKit exécute `stopstart` sur le module. Avec un handler spécial, le compteur `maxloop` n'est pas incrémenté. Utiliser la commande : `safekit incloop -m AM -i <handler name>` Cette commande incrémente le compteur et retourne quand la limite est atteinte.  Voir l'exemple décrit en section 15.4. Valeur par défaut : `stopstart`
`start_after=[nb polling cycles]`	Sans le paramètre `start_after`, la surveillance des processus est effective immédiatement. Sinon elle est retardée de `(n-1)*polltimer` secondes où : • `n` est la valeur de `start_after` • `polltimer` est le paramètre de `polling` d'`errd` (30 secondes par défaut) Par exemple si `start_after="3"`, la surveillance est retardée de 60 secondes ((3-1)*30). Le paramètre `start_after` est utile si le processus prend un certain temps à démarrer. Valeur par défaut : `0`
Paramètres avancées	
`atmax="-1"`	Nombre maximum de processus qui peuvent s'exécuter. Si le maximum est atteint, SafeKit déclenche l'action. Avec `atmax="0"`, une action est déclenchée à chaque démarrage du processus. Valeur par défaut : `-1` critère non pris en compte
`</errd>`	

13.9.4 <errd> Commandes

Si la commande est utilisée dans un script du module, alors la variable d'environnement SAFEMODULE est positionnée et le paramètre "-m *AM*" n'est pas nécessaire.

safekit -r errdpoll_running	Stocke son résultat dans le fichier <SAFEVAR>/errdpoll_reserrd (SAFEVAR = /var/safekit sur Linux ou c:\safekit\var sur Windows) avec une ligne pour chaque processus : <pid> <command name> <command full name and arguments list> (parent=<parent pid>) En Windows, le nom de la commande est affiché en minuscule. Utile pour déterminer le nom des processus à surveiller et leurs arguments pour une configuration <errd>
safekit errd disable "classname" -m *AM*	Suspend la surveillance des processus de la classe classname (pour le module applicatif *AM*). Doit être explicitement appelé dans les scripts stop_... avant d'arrêter l'application, pour des processus dans des classes différentes de prim, both, second, sec.
safekit errd enable "classname" -m *AM*	Redémarre la surveillance des processus de la classe classname (pour le module applicatif *AM*). Doit être explicitement appelé dans les scripts start_... après le démarrage de l'application, pour des processus dans des classes différentes de prim, both, second, sec.
safekit errd off -m *AM*	Suspend la surveillance des processus du module *AM* (sauf les processus SafeKit). Utile lorsque l'on stoppe manuellement l'application et que l'on ne veut pas de détection. Avec SafeKit < 8.2, utiliser safekit errd suspend -m *AM*
safekit errd on -m *AM*	Redémarre la surveillance des processus du module *AM* Avec SafeKit < 8.2, utiliser safekit errd resume -m *AM*
safekit errd list -m *AM*	Liste tous les processus du module *AM* surveillés incluant les processus SafeKit. La liste peut être également lue dans SAFEVAR/modules/*AM*/errdlist.

`safekit kill -name="process_name" [-argregex="…"] -level="kill_level"`	Le composant `<errd>` doit s'exécuter. Tue le(s) processus identifié(s) par le nom et les arguments. • `level="test"` : affiche seulement la liste des processus • `level="terminate"` : kill les processus en Windows • `level="9"` : envoie le signal SIGKILL aux processus en Linux • `level="15"` : envoie le signal SIGTERM aux processus en Linux • Exemples Windows ("class CatlRegExp" pour plus d'information): `safekit kill -name="notepad.exe" -argregex=".*myfile.*" -level="terminate"` `safekit kill -name="notepad.exe" -argregex="c:\\myrep\\myfile.*" -level="terminate"` • Exemples Linux ("man regex" pour plus d'information) : `safekit kill -name="vi" -argregex=".*myfile.*" -level="9"` `safekit kill -name="vi" -argregex="/myrep/myfile.*" -level="9"`

13.10 Checkers - <check>

SafeKit apporte des checkers qui testent un élément critique et affecte l'état d'une ressource du module en fonction du résultat du test. Sur détection d'erreur par un checker, la failover machine exécute une action sur le module d'après la règle de failover associée au checker. Pour leur description complète, voir la section 13.10.3.

Les checkers apportés par SafeKit sont :

- Section 13.11 « TCP checker - <tcp> »
- Section 13.12 « Ping checker - <ping> »
- Section 13.13 « Interface checker - <intf> ».
- Section 13.14 « IP checker - <ip> »
- Section 13.15 « Custom checker - <custom> »
- Section 13.16 « Module checker - <module> »
- Section 13.17 « Splitbrain checker - <splitbrain> »

13.10.1 <check> Exemple

Tous les checkers se définissent dans une seule section `<check>` :

```
<check>
```

```
  <!-- Insérer ci-dessous les tags <tcp> <ping> <intf> <ip> <custom> <module>
       <splitbrain> -->
</check>
```

13.10.2 <check> Syntaxe

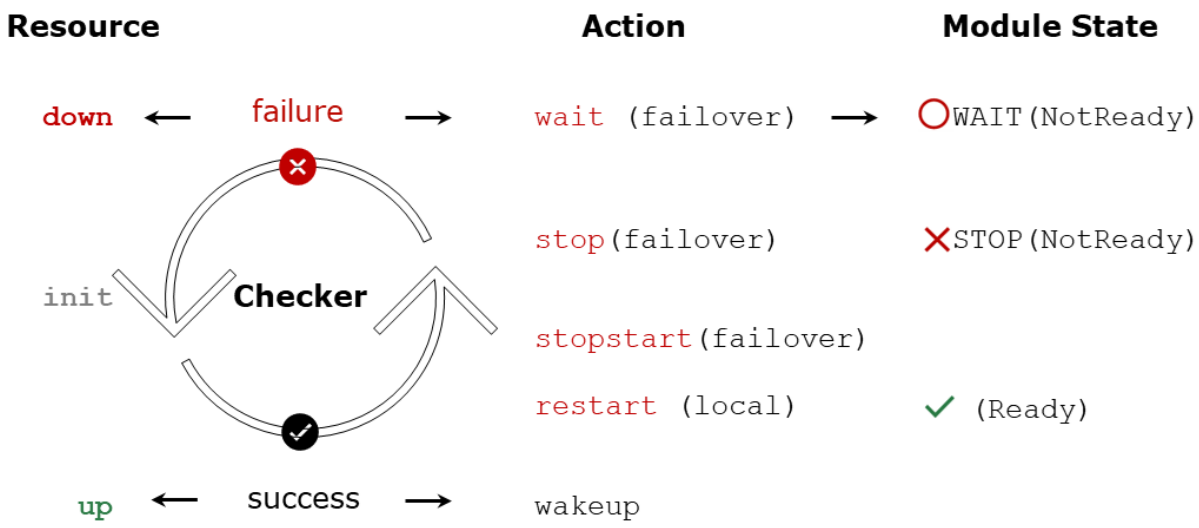
```
<check>
  <tcp …>
    <to …/>
  </tcp>
  …
  <ping …>
    <to …/>
  </ping>
  …
  <intf …>
    <to …/>
  </intf>
  …
  <ip …>
    <to …/>
  </ip>
  …
  <custom …/>
  …
  <module …>
    [<to …/>]
  </module>
  …
  <splitbrain …/>
</check>
```

Le tag `<check>` et son sous-arbre peuvent être entièrement modifiés dynamiquement.

13.10.3 <checker> Description

Un checker teste un élément critique (par défaut toutes les 10 secondes) et affecte l'état de la ressource associée, à `up` ou `down`, en fonction du résultat du test.

La failover machine évalue les règles de failover et exécute l'action associée au checker lorsque la ressource change d'état.



- L'état initial de la ressource est `init`. La failover machine laisse le module dans l'état `WAIT (Transient)`, tant qu'au moins une ressource utilisée par une règle avec action `wait` est dans l'état `init`.
- Si le test échoue, la ressource associée est mise dans l'état `down`. La règle de failover associée au checker définit quelle action exécuter dans ce cas. Les actions possibles sur le module sont `restart`, `stop`, `stopstart` ou `wait`.
  - L'action `restart` provoque le redémarrage local de l'application sans changement d'état du module.
  - Les actions `stop`, `stopstart` et `wait` consistent en l'arrêt du module, par conséquent de l'application, suivi d'un redémarrage automatique dans les cas `stopstart` et `wait`. L'arrêt du module peut entraîner le basculement sur l'autre nœud si celui-ci est `(Ready)`.
  - Lorsque l'action est `wait`, le module se bloque dans l'état `WAIT (NotReady)` tant que la ressource est `down`.

  Les actions `restart`, `stopstart` et `wait` incrémentent le compteur de détection d'erreurs. Lorsque celui-ci dépasse la limite `maxloop` dans l'intervalle de temps `loop_interval` (par défaut, à la 4ème détection d'erreur en 24h, voir section 13.2.3), le module est arrêté définitivement.
- Si le test réussi, la ressource associée est mise dans l'état `up`.

  Cela entraîne l'action implicite `wakeup` si l'action associée est `wait`. Le module sort de l'état `WAIT (NotReady)` et poursuit son démarrage normal.

La configuration du checker détermine :

- Le nom de la ressource associée
- Optionnellement, le nom de la règle de failover associée ainsi que l'action

13.10.3.1 Resource associée au checker

- L'état initial de la ressource est `init`

- Si le test échoue, la ressource associée est mise dans l'état `down`
- Si le test réussi, la ressource associée est mise dans l'état `up`

Pour la description des ressources, voir section 13.18.4.1.

Le nom de la ressource associée au checker est déterminé à partir de sa configuration :

- La classe de la ressource est la valeur du tag XML du checker : `tcp`, `ping`, `intf`, `ip`, `custom`, `module` ou `splitbrain`
- L'id de la ressource est la valeur de l'attribut `ident`

Par exemple, pour la configuration suivante d'un ping checker :

```
<check>
  <ping ident="testR2" action="wait">
    <to addr="R2"/>
  </ping>
</check>
```

la ressource associée se nomme **`ping.testR2`**.

La valeur courante de la ressource est visible :

- via la console web comme décrit en section 3.4.4.2
- avec la commande `safekit state -v -m` *`AM`* (où *`AM`* est le nom du module)

```
…
ping.testR2                down          yyyy-mm-dd
```

Les changements d'états de la ressource sont visibles dans le journal du module :

- via la console web comme décrit en section 3.4.4.1
- avec la commande `safekit logview -A -m` *`AM`* (où *`AM`* est le nom du module)

```
I | Resource ping.testR2 set to up by pingcheck
…
C | Resource ping.testR2 set to down by pingcheck
```

13.10.3.2 Règle de failover associée au checker

La règle de failover associée au checker définit quelle action exécuter lorsque sa ressource passe à `down`. Pour la description des règles de failover, voir section 13.18.4.2.

Les actions possibles sur le module sont `restart`, `stop`, `stopstart` ou `wait`.

Le règle de failover associée au checker est déterminé à partir de sa configuration :

- Les checkers `intf`, `ip`, `module` et `splitbrain` ont une règle par défaut prédéfinie qui s'applique à toutes les ressources de ce type :

```
/* rule for module checkers */
module_failure: if (module.? == down) then wait();

/* rule for interface checkers */
interface_failure: if (intf.? == down) then wait();

/* rule for ip checkers */
ip_failure: if (ip.? == down) then stopstart();
```

```
/* rules for splitbrain */
splitbrain_failure: if (splitbrain.uptodate == down) then wait();
```

- Les checkers `tcp`, `ping` et `custom` ont une règle générée avec la valeur de l'attribut `action` si celui est défini avec la valeur `stop`, `stopstart`, `restart` ou `wait`.

  Par exemple, pour la configuration suivante d'un ping checker

```
<check>
  <ping ident="testR2" action="wait">
    <to addr="R2"/>
  </ping>
</check>
```

  la règle générée est :

```
p_testR2 : if (ping.testR2 == down) then wait();
```

  Le nom de la règle a pour préfixe la première lettre du nom du checker (`t`, `p` ou `c`), suivie de _, puis de la valeur de l'attribut `ident`.

- Les checkers `tcp`, `ping` et `custom` n'ont pas de règle de failover si la valeur de l'attribut action dans sa configuration vaut `noaction`. Dans ce cas, l'utilisateur doit rajouter dans la configuration du module la règle de failover associée.

  Par exemple, pour la configuration suivante d'un custom checker, la règle de failover est ajoutée explicitement.

```
<check>
  <custom ident="checkfile" exec="checker.ps1"
          arg="c:\safekit\checkfile" when="prim" action="noaction"/>
</check>

<failover>
  <![CDATA[
    checkfile_failure: if( custom.checkfile == down ) then restart();
  ]]>
</failover>
```

Lorsque la règle de failover est activée, celle-ci est visible :

- via la console web dans l'état détaillé du module décrit en section 3.4.2.2
- par un message dans le journal du module semblable au suivant :

  heart | C | Action **wait** d'après la règle de failover **p_testR2**

  Le journal du module pouvant être visualisé :

  - via la console web comme décrit en section 3.4.4.1
  - avec la commande `safekit logview -A -m` *AM* (où *AM* est le nom du module)

13.11 TCP checker - <tcp>

Par défaut, il y a une action `restart` du module lorsque le tcp checker détecte un échec de connexion au service TCP.

Depuis SafeKit 8.2.3, l'action peut être configurée avec l'attribut `action` du tag `<tcp>`.

Insérer le tag <tcp> dans la section <check> si celle-ci est déjà définie.

13.11.1 <tcp> Exemple

```
<check>
  <tcp ident="R1test" when="prim" action="restart" >
    <to addr="R1" port="80"/>
  </tcp>
</check>
```

- La ressource associée au checker se nomme tcp.R1test (préfixe tcp.)
- La règle de failover générée, qui effectue un restart lorsque la ressource passe à down, se nomme t_R1test (préfixe t_). Elle est équivalente à :

```
t_R1test: if (tcp.R1test == down) then restart();
```

Pour une description des checkers, voir la section 13.10.3.

Voir aussi l'exemple en section 15.5. Elle présente la configuration via la console web ainsi que le userconfig.xml correspondant.

13.11.2 <tcp> Syntaxe

```
<tcp
    ident="tcp_checker_name"
    when="prim|second|both|pre"
    [action=" stop|stopstart|restart|wait|noaction"]
>
  <to
     addr="IP_address" or "name_to_check"
     port="TCP_port_to_check"
     [interval="10"]
     [timeout="5"]
   />
</tcp>
```

Depuis SafeKit 8.2.3, utilisez l'attribut action pour définir l'action à effectuer sur détection d'erreur par le tcp checker.

Avant SafeKit 8.2.3, l'action était statique et définie par la règle de failover par défaut qui s'applique à toutes les ressources de classe tcp :

```
tcp_failure: if (tcp.? == down) then restart();
```

13.11.3 <tcp> Attributs

<tcp	Positionner autant de sections <tcp> qu'il y a de checkers <tcp>.
ident="*tcp_checker_name*"	Nom du checker TCP. Il définit la ressource associée au checker : **tcp**.*tcp_checker_name* (préfixe tcp.)

when="prim|second|both" [action="stop|stopstart|restart|noaction"]	Utiliser cette valeur pour tester un service TCP interne à l'application, une fois celle-ci démarrée : • when="prim" pour une module miroir Le checker est démarré après/arrêté avant, l'exécution des scripts start_prim/stop_prim • when="both" pour une module ferme Le checker est démarré après/arrêté avant, l'exécution des scripts start_both/stop_both • when="second" pour une module miroir Le checker est démarré après/arrêté avant, l'exécution des scripts start_second/stop_second Depuis SafeKit 8.2.3, vous pouvez configurer l'action à effectuer en cas de détection d'erreur avec : • action="stop|stopstart|restart" • stop, stopstart ou restart du module. Le nom de la règle de failover associée est **t**_*tcp_checker_name* (préfixe t_) • action="noaction" pas de règle de failover générée. L'action doit être écrite explicitement dans le tag <failover> (voir section 13.18). Valeur par défaut : action="restart"
when="pre" action="wait|noaction"	Utiliser cette valeur pour tester un service TCP externe à l'application avant son démarrage : • when="pre" Le checker est démarré après/arrêté avant, l'exécution des scripts prestart/poststop Depuis SafeKit 8.2.3, vous pouvez configurer l'action à effectuer en cas de détection d'erreur avec : • action="wait" wait du module. Le nom de la règle de failover associée est t_*tcp_checker_name* (préfixe t_) • action="noaction" pas de règle de failover générée. L'action doit être écrite explicitement dans le tag <failover> (voir section 13.18).
<to	

addr="*IP_@*" or "*name*"	Adresse IP ou nom à checker (ex : 127.0.0.1 pour un service local). Adresse IPv4 ou IPv6.
port="*value*"	Port TCP port à checker
[interval="10"]	Temps, en secondes, entre 2 pollings. Valeur par défaut : 10 secondes
[timeout="5"]	Délai en secondes pour l'acceptation de la connexion. Valeur par défaut : 5 secondes
</tcp>	

13.12 Ping checker - <ping>

Par défaut, il y a un wait du module lorsque le ping checker détecte un échec de ping sur l'équipement.

Depuis SafeKit 8.2.3, l'action peut être configurée avec l'attribut action du tag <tcp>.

Insérer le tag <ping> dans la section <check> si celle-ci est déjà définie.

13.12.1 <ping> Exemple

```
<check>
  <ping ident="testR2" action="wait">
    <to addr="R2"/>
  </ping>
</check>
```

- La ressource associée au checker se nomme ping.testR2 (préfixe ping.)
- La règle de failover générée, qui effectue un wait lorsque la ressource passe à down, se nomme p_testR2 (préfixe p_) et est équivalente à :

```
p_testR2: if (ping.testR2== down) then wait();
```

Pour une description des checkers, voir la section 13.10.3.

Voir aussi l'exemple en section 15.6. Elle présente la configuration via la console web ainsi que le userconfig.xml correspondant.

13.12.2 <ping> Syntaxe

```
<ping
    ident="ping_checker_name"
    [when="pre|prim|second|both"]
    [action="wait|stop|stopstart|restart|noaction"]

>
  <to
     addr="IP_address" or "name_to_check"
```

```
    [interval="10"]
    [timeout="5"]
  />
</ping>
```

Depuis SafeKit 8.2.3, utilisez l'attribut `action` pour définir l'action à effectuer sur détection d'erreur par le ping checker.

Avant SafeKit 8.2.3, l'action était statique et définie par la règle de failover par défaut :

```
ping_failure: if (ping.? == down) then wait();
```

13.12.3 <ping> Attributs

`<ping`	Mettre autant de section <ping> qu'il y a de ping checkers.
`ident="`*`ping_checker_name`*`"`	Nom du checker ping Il définit la ressource associée au checker : **`ping`**.*`ping_checker_name`* (préfixe `ping.`)
`when="pre"` `action="wait|noaction"`	Utiliser cette valeur pour tester un équipement externe à l'application, avant son démarrage : • `when="pre"` Le checker est démarré après/arrêté avant, l'exécution des scripts `prestart/poststop` Depuis SafeKit 8.2.3, vous pouvez configurer l'action à effectuer en cas de détection d'erreur avec : • `action="wait"` • `wait` du module. Le nom de la règle de failover associée est **`p`**_*`ping_checker_name`* (préfixe `p_`) • `action="noaction"` pas de règle de failover générée. L'action doit être écrite explicitement dans le tag `<failover>` (voir section 13.18). Valeur par défaut : `when="pre" action="wait"`
`when="prim|second|both"` `[action="stop|stopstart|restart|noaction"]`	Utiliser cette valeur pour tester un équipement après le démarrage de l'application : • `when="prim"` pour une module miroir Le checker est démarré après/arrêté avant, l'exécution des scripts `start_prim/stop_prim` • `when="both"` pour une module ferme Le checker est démarré après/arrêté avant, l'exécution des scripts `start_both/stop_both`

	• when="second" pour une module miroir Le checker est démarré après/arrêté avant, l'exécution des scripts start_second/stop_second Depuis SafeKit 8.2.3, vous pouvez configurer l'action à effectuer en cas de détection d'erreur avec : • action="stop|stopstart|restart" • stop, stopstart ou restart du module. Le nom de la règle de failover associée est **p**_*ping_checker_name* (préfixe p_) • action="noaction" pas de règle de failover générée. L'action doit être écrite explicitement dans le tag <failover> (voir section 13.18).
<to	
addr="*IP_@ or name*"	Adresse IP ou nom à checker Adresse IPv4 ou IPv6.
[interval="10"]	Intervalle de temps, en secondes, entre 2 pollings. Valeur par défaut : 10 secondes
[timeout="5"]	Délai en secondes sur la réponse au ping. Valeur par défaut : 5 secondes
</ping>	

13.13 Interface checker - <intf>

Par défaut, il y a un wait du module lorsque l'interface checker détecte une erreur sur l'interface.

Insérer le tag <intf> dans la section <check> si celle-ci est déjà définie.

13.13.1 <intf> Exemple

```
<check>
  <intf ident="test_eth0">
    <to local_addr="192.168.1.10"/>
  </intf>
</check>
```

- La ressource associée au checker se nomme intf.test_eth0 (préfixe intf.)
- La règle de failover, qui effectue un wait lorsqu'une ressource de classe intf passe à down, est statique et définie par la règle de failover par défaut :

```
interface_failure: if (intf.? == down) then wait();
```

Pour une description des checkers, voir la section 13.10.3.

Voir aussi l'exemple en section 15.10. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.13.2 <intf> Syntaxe

```
<intf
    ident="intf_checker_name"
    [when="pre"]

>
  <to
     local_addr="interface_physical_IP_address"/>
</intf>
```

13.13.3 <intf> Attributs

`<intf`	Les sections `<intf>` sont automatiquement générées lorsque `<interface check="on">` est positionné (voir section 13.5).
`ident="`*`intf_checker_name`*`"`	Le nom de l'interface checker (adresse IP du réseau). Il définit la ressource associée au checker : **`intf.`***`intf_checker_name`* (préfixe `intf.`)
`[when="pre"]`	Valeur fixe • `when="pre"` Le checker est démarré après/arrêté avant, l'exécution des scripts `prestart/poststop` En cas de détection d'erreur, l'action est `wait`. Le nom de la règle de failover, **`interface_failure`**, est statique et prédéfinie.
`<to local_addr="`*`IP_@`*`" />`	Adresse IP associée à l'interface à tester. Adresse IPv4 ou IPv6.
`</intf>`	

13.14 IP checker - <ip>

Par défaut, il y a un `stopstart` du module lorsque l'ip checker détecte que l'adresse IP n'est pas configurée localement. En Windows, il détecte en plus les conflits sur cette adresse.

Insérer le tag `<ip>` dans la section `<check>` si celle-ci est déjà définie.

13.14.1 <ip> Exemple

```
<check>
  <ip ident="ip_check" >
```

```
    <to  addr="192.168.1.10" />
  </ip>
</check>
```

- La ressource associée au checker se nomme `ip.ip_check`
- La règle de failover, qui effectue un `stopstart` lorsqu'une ressource de classe `ip` à `down`, est statique et définie par la règle de failover par défaut :

```
ip_failure: if (ip.? == down) then stopstart();
```

Pour une description des checkers, voir section 13.10.3.

Voir aussi l'exemple en section 15.11. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.14.2 <ip> Syntaxe

```
  <ip
    ident="ip_checker_name"
    [when="prim"|"both"]
  >
    <to
     addr="IP_address" or "name_to_check"
     [interval="10"]
     />
  </ip>
```

13.14.3 <ip> Attributs

`<ip`	Les sections `<ip>` sont automatiquement générées lorsque `<virtual_addr check="on">` est positionné (voir section 13.5).
`ident="`*`ip_checker_name`*`"`	Nom du checker ip Il définit la ressource associée au checker : **`ip.`***`ip_checker_name`* (préfixe `ip.`)
`[when="prim"|"both"]`	Valeur par défaut • `when="prim"` pour une module miroir Le checker est démarré après/arrêté avant, l'exécution des scripts `start_prim/stop_prim` • `when="both"` pour une module ferme Le checker est démarré après/arrêté avant, l'exécution des scripts `start_both/stop_both` En cas de détection d'erreur, l'action est `stopstart`. Le nom de la règle de failover, **`ip_failure`**, est statique et prédéfinie.
`<to`	

addr="*IP_@ or name*"	Adresse IP locale ou nom DNS à tester. Adresse IPv4 ou IPv6.
[interval="10"]	Intervalle de temps, en secondes, entre 2 tests. Valeur par défaut : 10 secondes
</ip>	

13.15 Custom checker - <custom>

Un checker customisé est un exécutable (script ou binaire) que vous développez pour tester une ressource, l'application…. Il s'agit d'une boucle qui effectue un test suivant une période adéquate. Son rôle est de positionner la ressource associée au checker à "up" ou "down". Puis, une règle de failover décide l'action à exécuter sur le module lorsque la ressource est down.

Depuis SafeKit 8, l'action peut être configurée avec l'attribut action du tag <custom>.

Insérer le tag <custom> dans la section <check> si celle-ci est déjà définie.
Définir en plus le checker customisé.

13.15.1 <custom> Exemple

- Exemple avec action!="noaction"

```
<check>
  <custom ident="AppChecker" when="prim" exec="mychecker" action="stopstart"/>
</check>
```

  - La ressource associée au checker se nomme custom.AppChecker
  - La règle de failover, qui effectue un stopstart lorsque la ressource passe à down, se nomme c_AppChecker et est équivalente à :

```
c_AppChecker: if (custom.AppChecker == down) then stopstart();
```

- Exemple avec action="noaction"

```
<check>
  <custom ident="AppChecker" when="prim" exec="mychecker" action="noaction"/>
</check>
```

  Aucune règle de failover n'est générée. L'utilisateur a la possibilité d'en définir une explicitement dans le tag <failover>. Par exemple :

```
…
<failover>
  <![CDATA[
    custom_failure: if (custom.AppChecker == down) then stopstart();
  ]]>
</failover>
```

Pour une description des checkers, voir section 13.10.3.

Pour un exemple complet, voir la section 15.7. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

Dans SafeKit < 8, l'attribut `action` n'existait pas et l'action était configurée en définissant une règle de failover dans le tag `<failover>` comme l'exemple ci-dessus. C'est pourquoi, la valeur par défaut de l'attribut `action` est équivalente à `"noaction"` est afin de préserver la compatibilité ascendante avec une ancienne configuration.

13.15.2 <custom> Syntaxe

```
<custom
   ident="custom_checker_name"
   when="pre|prim|second|both"
   exec="executable_path"
   arg="executable_arguments"
   action="wait"|"stop"|"stopstart"|"restart"|"noaction"
/>
```

13.15.3 <custom> Attributs

`<custom`	Positionner autant de sections <custom> qu'il y a de checkers customisés.
`ident="`*`custom_checker_name`*`"`	Nom du checker customisé Il définit la ressource associée au checker : **`custom.`***`custom_checker_name`* (préfixe `custom.`) Le checker customisé positionne sa ressource avec la commande `safekit set -r custom.`*`custom_checker_name`*` -v up|down`.
`when="pre"` `action="wait"|"noaction"`	Utiliser cette valeur pour tester un composant externe à l'application, avant son démarrage : • `when="pre"` Le checker est démarré après/arrêté avant, l'exécution des scripts `prestart/poststop` Depuis SafeKit 8, vous pouvez configurer l'action à effectuer en cas de détection d'erreur avec : • `action="wait"` `wait` du module. Le nom de la règle de failover associée est **`c_`***`custom_checker_name`* (préfixe `c_`) • `action="noaction"` pas de règle de failover générée. L'action doit être écrite explicitement dans le tag `<failover>` (voir section 13.18).

when="prim"|"second"|"both" action="stop"|"stopstart"|"restart"|"noaction"	Utiliser cette valeur pour tester un composant après le démarrage de l'application : • when="prim" pour une module miroir Le checker est démarré après/arrêté avant, l'exécution des scripts start_prim/stop_prim • when="both" pour une module ferme Le checker est démarré après/arrêté avant, l'exécution des scripts start_both/stop_both • when="second" pour une module miroir Le checker est démarré après/arrêté avant, l'exécution des scripts start_second/stop_second Depuis SafeKit 8, vous pouvez configurer l'action à effectuer en cas de détection d'erreur avec : • action="stop|stopstart|restart" stop, stopstart ou restart du module. Le nom de la règle de failover associée est **c**_*custom_checker_name* (préfixe c_) • action="noaction" pas de règle de failover générée. L'action doit être écrite explicitement dans le tag <failover> (voir section 13.18).
exec="*executable_path*"	Défini le chemin de l'exécutable du checker customisé (un script ou un binaire). Lorsque le chemin est relatif, le custom checker est recherché dans SAFEUSERBIN. Mettre dans ce cas votre binaire dans SAFE/modules/*AM*/bin/ (pour plus d'information, voir section 10.1). Nous conseillons un chemin relatif avec un exécutable dans le module. • En Windows, l'exécutable peut être un binaire ou un script ps1, vbs ou cmd. • En Linux, l'exécutable peut être un binaire ou un script shell.
arg="*executable_arguments*"	Défini les arguments à passer au checker customisé.

13.16 Module checker - <module>

Par défaut, il y a un wait du module lorsque le module checker détecte l'indisponibilité d'un autre module SafeKit. Le checker de module effectue également une action stopstart lorsqu'il détecte que le module externe a été redémarré (soit par un restart, un stopstart ou à la suite d'un basculement).

Le checker de module récupère l'état du module en se connectant au service web de SafeKit qui s'exécute sur le serveur sur lequel le module est activé (voir section 10.7).

Insérer le tag `<module>` dans la section `<check>` si celle-ci est déjà définie.

13.16.1 <module> Exemple

- Exemple utilisant la configuration par défaut du service web de SafeKit (protocole : HTTP, port : 9010) :

```
<check>
  <module name="mysql">
    <to  addr="172.24.190.21" port="9010"/>
  </module>
</check>
```

`mysql` est le nom du module externe et `172.24.190.21` est son IP virtuelle.

  - La ressource associée au checker se nomme `module.mysql_172.24.190.21`
  - La règle de failover, qui effectue un `wait` lorsqu'une ressource de classe `module` passe à `down`, est statique et définie par la règle de failover par défaut :

```
module_failure: if (module.? == down) then wait();
```

- Pour le même exemple en utilisant la configuration sécurisée du service web de SafeKit (protocole : HTTPS, port : 9453) :

```
<check>
  <module name="mysql">
    <to  addr="172.24.190.21" port="9453" secure="on"/>
  </module>
</check>
```

Pour une description des checkers, voir section 13.10.3.

Pour d'autres exemples, voir la section 15.9. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.16.2 <module> Syntaxe

```
<module
  [ident="module_checker_name"]
  name="external_module_name">
  [<to
   addr="IP_@ or name the Safekit server running the external module"
   port="port of the SafeKit web server"
   [interval="10"]
   [timeout="5"]
   />]
</module>
```

13.16.3 <module> Attributs

`<module`	Positionner autant de sections `<module>` qu'il y a de checkers de module.
`name="`*`external_module_name`*`"]`	Nom du checker de module.

[ident="*module_checker_name*"]	Nom du module externe à checker. Il définit la ressource associée au checker : **module.***module_checker_name* (préfixe module.) Si cet attribut n'est pas renseigné, le nom de la ressource est construit à partir des attributs name et addr : **module.***external_module_name_address_or_name*
[<to	Définition du/des serveurs exécutant le module externe. Par défaut, le serveur local.
addr="*address_or_name*"	Adresse IP ou nom pour accéder au module externe. Adresse IPv4 ou IPv6.
port="*port du service web de SafeKit*"	Port du service web SafeKit pour le checking. 9010 pour HTTP ; 9453 pour HTTPS.
[interval="10"]	Intervalle de temps, en secondes, entre 2 pollings. Valeur par défaut : 10 secondes
[timeout="5"]	Délai en secondes pour la réception d'une réponse à la requête check. Valeur par défaut : 5 secondes
[secure="on"|"off"]	Utilisation du protocole HTTP (secure="off") ou HTTPS (secure="on") Par défaut : off
/>]	
</module>	

13.17 Splitbrain checker - <splitbrain>

SafeKit offre un checker de split-brain à utiliser pour des architectures miroir. Le split-brain est une situation où, à la suite d'une panne matérielle ou logicielle, les deux nœuds SafeKit deviennent primaires, chaque nœud supposant que l'autre ne fonctionne plus. Ceci implique que l'application tourne sur les deux nœuds et, lorsque la réplication est active, les données peuvent se trouver dans un état incohérent.

Pour prévenir ce phénomène, le checker de split-brain, sur détection d'isolation réseau entre les serveurs, sélectionne un unique nœud pour devenir primaire. L'autre nœud devient non à jour et se bloque dans l'état WAIT :

- Jusqu'à ce qu'il reçoive à nouveau les heartbeats de l'autre serveur

  Ou

- Si l'administrateur le juge opportun et s'assure que l'autre serveur n'est plus dans l'état primaire, il peut forcer son démarrage en primaire (par l'exécution des commandes `safekit stop -m AM` puis `safekit prim -m AM`).

L'élection du serveur primaire repose sur le ping d'un composant externe, appelé **witness**. Le réseau doit être configuré de telle manière qu'en cas d'isolation réseau, un seul des deux serveurs a accès au witness (c'est celui-ci qui sera élu primaire). Dans le cas contraire, les deux nœuds deviendront primaires.

- Le ping entre les 2 nœuds et avec le witness doit être ouvert

- Depuis SafeKit 8.2.1, il est possible de définir plusieurs witness. Cela permet de tolérer la panne d'un witness, au moins un devant être accessible.

Insérer le tag `<splitbrain>` dans la section `<check>` si celle-ci est déjà définie.

13.17.1 <splitbrain> Exemple

```
<check>
  <splitbrain ident="witness" exec="ping" arg="192.168.1.100 192.168.2.120" />
</check>
```

- La ressource associée au checker se nomme `splitbrain.witness`
- En cas d'isolation réseau entre les nœuds, le split-brain checker affecte la ressource `splitbrain.uptodate` à `up` ou `down` en accord avec l'accès au witness.
- La règle de failover, qui effectue un `wait` lorsque la ressource `splitbrain.uptodate` passe à `down`, est statique et prédéfinie :

```
splitbrain_failure: if (splitbrain.uptodate == down) then wait();
```

Voir aussi l'exemple en section 15.8. Elle présente la configuration via la console web ainsi que le `userconfig.xml` correspondant.

13.17.2 <splitbrain> Syntaxe

```
  <splitbrain
     ident="witness"
     exec="ping"
     arg="witness1_IP_name  witness2_IP_name"
  />
```

13.17.3 <splitbrain> Attributs

`<splitbrain`	Une seule section `<splitbrain>`.
`ident="witness_name"`	Nom de checker split-brain

	Il définit la ressource associée au checker : **splitbrain**.*witness_name* (préfixe splitbrain.) Elle représente l'état du ou des witness. La ressource est affectée à : • up, si au moins un des witness répond • down, si tous les witness ne répondent pas
[when="pre"]	Valeur fixe • when="pre" Le checker est démarré après/arrêté avant, l'exécution des scripts prestart/poststop Sur détection de split-brain : • Le nœud qui a accès au witness (splitbrain.witness_name="up") affecte la ressource splitbrain.uptodate à up et devient le primaire. • L'autre serveur qui n'a pas accès au witness (splitbrain.witness_name="down") affecte la ressource splitbrain.uptodate à down. Cela déclenche l'action wait de la règle de failover statique et prédéfinie, nommée **splitbrain_failure**.
exec="ping"	Valeur fixe. Utilise ping pour tester l'accessibilité au witness et affecte la ressource **splitbrain**.*witness_name*
arg="*witness1_IP_name witness2_IP_name*"	Liste des adresses IP ou nom des witness Adresse IPv4 ou IPv6. La définition de plusieurs witness est supportée depuis SafeKit 8.2.1.
</splitbrain>	

13.18 Failover machine - <failover>

SafeKit apporte des checkers qui testent un élément critique et affecte l'état de la ressource associée en fonction du résultat du test. Sur détection d'erreur par un checker, la failover machine exécute une action sur le module d'après la règle de failover associée au checker. Pour leur description complète, voir la section 13.10.

Certains composants SafeKit (<heart>, <rfs>, <vipd>, <errd>) gèrent leurs propres ressources et apportent leurs propres règles de failover. Ces dernières ne doivent pas être modifiées ni supprimées, sous peine d'un comportement anormal de SafeKit.

La failover machine évalue régulièrement (par défaut toutes les 5 secondes) l'état global de toutes les ressources et applique une action en fonction des règles de failover qui sont vraies.

Dans un module ferme, la failover machine ne fonctionne que sur les états locaux des ressources alors que dans un module miroir, une règle peut tester une ressource locale ou distante.

Comme les états des ressources transitent par les voies de heartbeat, il est conseillé d'en avoir plusieurs (voir section 13.3).

Les règles de failover peuvent être écrites dans un langage simple propre à SafeKit ou dans le langage Lua via l'appel de fonctions SafeKit.

13.18.1 <failover> Exemple

Les exemples de règles écrites dans cette section se rajoutent aux règles par défaut ou générées d'après la configuration des checkers.

- Exemple d'ajout d'une règle écrite avec le langage de la failover machine

```
<failover>
   <![CDATA[
     custom_failure: if (custom.AppChecker == down) then stopstart();
   ]]>
</failover>
```

- Exemple d'ajout d'une règle avec le langage Lua et l'appel de la fonction `if_then`

  Le préfixe « `--Lua Rules` » indique que la suite de la section doit être interprétée avec l'interpréteur Lua.

```
<failover>
   <![CDATA[
     --Lua Rules
     Rules = Rules +
     {
custom_failure=if_then("custom.AppChecker","down",Action.stopstart),_group="ch
ecker"}
   ]]>
</failover>
```

- Exemple de règle pour désactiver la règle par défaut nommée `ip_failure` et ajouter la règle `allip_failure`

```
<failover>
<![CDATA[
  --Lua Rules
  Rules.disable("ip_failure")
  -- Ajouter ici les éventuelles règles lua destinées à remplacer les règles
en question, ou écrire les règles legacy dans une autre section CDATA
]]>
<![CDATA[
   allip_failure: if (ip.* == down) then stopstart();
]]>
</failover>
```

Utiliser une section `<![CDATA[ … ]]>` différente pour chaque langage.

13.18.2 <failover> Syntaxe

```
<failover [extends="yes"] [period="5000"] [handle_time="15000"]>
<![CDATA[
  label: if (expression) then action;
```

```
  …
]]>
</failover>
```

Le tag `<failover>` et son sous-arbre **ne peuvent pas** être modifiés dynamiquement.

13.18.3 <failover> Attributs

`<failover`	
`[extends="yes"|"no"]`	• `extends="yes"` les nouvelles règles de failover étendent les règles par défaut (voir section 13.18.4). • `extends="no"` les règles par défaut sont écrasées (à éviter) Valeur par défaut : `yes`
`[period="5000"]`	Période entre 2 évaluations. Valeur par défaut : `5000` millisecondes (5 secondes)
`[handle_time="15000"]`	Une action (`stop()` | `stopstart()` | `wait()` | `restart()` | `swap`()) doit être stable pendant `handle_time` (en millisecondes) avant d'être appliquée. Valeur par défaut : `15000` millisecondes (15 secondes). `handle_time` doit être un multiple de `period`.

13.18.4 <failover> Description

13.18.4.1 Ressources du module

La syntaxe pour désigner les ressources est la suivante :

```
resource ::= [local. | remote.] 0/1resource_class.resource_id     (default: local)
resource_class ::= ping | intf | tcp | ip | custom | splitbrain | module |
heartbeat | rfs
resource_id ::= * | ? | name
resource_state ::= init | down | up | unknown
```

`init`	État spécial d'initialisation tant que le checker n'a pas démarré. Si une ressource dans l'état init est testé dans une règle de failover, cette règle n'est pas évaluée tant que la ressource est dans l'état init.
`up`	État OK
`down`	État KO
`unknown`	État inconnu pour une ressource distante (module arrêté sur l'autre nœud)

13.18.4.2 Règles de failover

SafeKit apporte des règles de failover par défaut et des règles générées automatiquement en fonction de la configuration des checkers du module. L'utilisateur peut également écrire ses propres règles.

Règles de failover par défaut

Les règles de failover par défaut pour les checkers (`module`, `intf`, `ip`, `splitbrain`) sont :

```
<failover>
<![CDATA[
  /* rule for module checkers */
  module_failure: if (module.? == down) then wait();

  /* rule for interface checkers */
  interface_failure: if (intf.? == down) then wait();

  /* rule for ip checkers */
  ip_failure: if (ip.? == down) then stopstart();

  /* rules for splitbrain */
  splitbrain_failure: if (splitbrain.uptodate == down) then wait();
]]>
</failover>
```

Il existe en plus :

- Des règles de failover dédiées à la gestion de la réplication, des heartbeats….
- La règle `Implicit_wakeup` qui est appliquée lorsqu'aucune action `wait` n'est applicable. Elle exécute l'action `wakeup`.

Depuis SafeKit 7.5, les règles de failover utilisent une nouvelle syntaxe basée sur le langage Lua.

Règles de failover générées

Les checkers `tcp`, `ping` et `custom` ont une règle de failover automatiquement générée lorsque la valeur de l'attribut `action` est définie à `stop`, `stopstart`, `restart` ou `wait`. Le nom de la règle commence par la première lettre du nom du checker (`t`, `p` ou `c`), suivie de _, puis de la valeur de l'attribut `ident` (par exemple, `p_router`, `t_service`, `c_app`).

Règles de failover configurées

L'utilisateur peut aussi définir ses propres règles de failover dans la section `<failover><![CDATA[ … ]]></failover>`. Par défaut, celles-ci s'ajoutent aux règles par défaut et générées.

Voir les exemples dans la section 13.18.1.

Les règles de failover peuvent être écrites en suivant une des syntaxes suivantes :

- Langage de la failover machine

```
label: if ( expression ) then action;

label ::= string
```

```
action ::= stop() | stopstart() | wait() | restart() | swap()

expression ::= ( expression )
| ! expression
| expression && expression
| expression || expression
| expression == expression
| expression != expression
| resource ::= [local. | remote.] 0/1resource_class.resource_id
| resource_state
```

- Langage Lua
  - Appel de la fonction `if_then` pour définir une nouvelle règle

```
--Lua Rules
Rules = Rules +
{ label=if_then("resource","resource_state",action),_group="checker" }
```

```
label ::= string

action ::= Action.stop | Action.stopstart | Action.wait | Action.restart
| Action.swap

| resource ::= resource_class.resource_id
| resource_state
```

  - Appel de la fonction `Rules.disable` pour désactiver une règle à partir de son label

```
--Lua Rules
Rules.disable("failover_rule_label")
```

Utiliser une section `<![CDATA[ … ]]>` différente pour chaque langage.

13.18.4.3 Actions

Les actions `restart()`, `stopstart()`, `stop()`, `swap()` sont équivalentes aux commandes de contrôle (avec le paramètre `-i identity` en plus) décrites en section 9.3.

`maxloop / loop_interval / automatic_reboot` s'appliquent si le paramètre `-i identity` est passé aux commandes (voir section 13.2). Ce qui est le cas lorsque les commandes sont appelées par la failover machine ou par un checker.

14. Scripts du module pour la configuration du module

Des exemples de scripts sont donnés en section 15.

Pour activer l'appel des scripts du module, le tag `<user>` doit être présent dans `userconfig.xml` (voir section 13.7).

Les scripts doivent être des exécutables :

- en Windows, un exécutable avec l'extension et le type : `.cmd`, `.vbs`, `.ps1`, `.bat` ou `.exe`
- en Linux, tout type d'exécutable

Chaque fois que vous modifiez les scripts, vous devez réappliquer la configuration sur les serveurs (avec la console ou la commande SafeKit).

Au moment de la configuration, les scripts sont copiés de `SAFE/modules/AM/bin` dans le répertoire d'exécution `SAFE/private/modules/AM/bin` (=`SAFEUSERBIN`, ne pas modifier les scripts à cet endroit).

14.1 Liste des scripts

Ci-dessous la liste des scripts pouvant être définis par l'utilisateur. Les scripts essentiels sont les scripts start/stop qui démarrent et arrêtent l'application au sein du module.

14.1.1 Scripts start/stop

`start_prim` `stop_prim`	**Scripts pour un module miroir.** Démarre l'application sur le serveur `ALONE` ou `PRIM`
`start_both` `stop_both`	**Scripts pour un module ferme.** Démarre l'application sur tous les serveurs `UP` de la ferme. Dans le cas spécial où ils existent dans un module miroir, ils sont exécutés dans les états `PRIM`, `SECOND` ou `ALONE`
`start_second` `stop_second`	**Scripts spéciaux pour un module miroir.**

	Exécutés sur le serveur `SECOND`
`start_sec` `stop_sec`	**Scripts spéciaux pour un module miroir.** Voir l'automate d'exécution des scripts décrit en section 14.4
`stop_[both, prim, second, sec] [force]`	**Scripts pour tous les modules.** Ces scripts sont appelés 2 fois : une fois pour un arrêt propre (sans le paramètre `force` en premier argument) et une fois pour un arrêt forcé (avec le paramètre `force` en premier argument)
`prestart` `poststop`	**Scripts pour tous les modules.** Exécutés au tout début du démarrage du module et à la fin. Par défaut, `prestart` contient `stop_sec`, `stop_second`, `stop_prim`, `stop_both` pour arrêter l'application avant de démarrer le module sous contrôle SafeKit
`transition`	**Scripts pour tous les modules.** Ce script est appelé sur changement d'état décrit en section 14.4

14.1.2 Autres scripts

`config`	`config` est appelé lors de l'exécution de la commande `safekit config -m` *AM*. Vous pouvez exécuter dans ce script des commandes de configuration applicative.
`deconfig`	`deconfig` est appelé lors de l'exécution de la commande `safekit deconfig -m` *AM*, celle-ci étant automatiquement effectuée lors de la désinstallation du module Dans ce script, vous devez « défaire » les actions effectuées dans le script `config`.
`confcheck`	`confcheck` est appelé lors de l'exécution de la commande `safekit confcheck -m` *AM*. Vous pouvez exécuter dans ce script des opérations de contrôle de changement de configuration de l'application.
`state`	`state` est appelé lors de l'exécution de la commande `safekit state -v -m` *AM*.
`level`	`level` est appelé lors de l'exécution de la commande `safekit level -m` *AM* command.

14.2 Variables d'environnement et arguments passés aux scripts

Tous les scripts sont appelés avec 3 paramètres :

- l'état courant (`STOP`, `WAIT`, `ALONE`, `PRIM`, `SECOND`, `UP`)
- le prochain état (`STOP`, `WAIT`, `ALONE`, `PRIM`, `SECOND`, `UP`)
- l'action (`start`, `stop`, `stopstart` ou `stopwait`)

L'argument `stopwait` est passé lors de l'exécution de l'action `wait` provoquée par un checker.

- Les scripts de type `stop` sont appelés 2 fois :
- une première fois pour un arrêt propre de l'application
- une deuxième fois avec l'argument `force` pour un arrêt forcé (avec `force` comme premier argument)

Les variables d'environnement utilisables à l'intérieur des scripts sont :

- `SAFE`, `SAFEMODULE`, `SAFEBIN`, `SAFEUSERBIN`, `SAFEUSERVAR` (voir section 10.1)

La définition de `SAFEMODULE`, qui contient le nom du module, permet d'omettre l'option `-m` *AM* avec la commande `safekit` si celle-ci doit s'appliquer au module `SAFEMODULE`.

- toutes les variables définies dans le tag `<user>` de `userconfig.xml` (voir section 13.7).

Pour un exemple avec les variables d'environnement, voir la section 15.3.

14.3 Sortie des scripts

14.3.1 Sortie dans le journal du script

Par défaut (`logging="userlog"` dans le tab <user> de `userconfig.xml`), les sorties stdout et stderr du script sont redirigées dans le fichier `SAFEVAR/modules/`*AM*`/userlog_<year>_<month>_<day>T<time>_<script name>.ulog` où :

- `SAFEVAR=C:\safekit\var` en Windows et `/var/safekit` en Linux
- *AM* est le nom du module
- `<year>_<month>_<day>T<time>` sont la date et heure d'exécution du script
- `<script name>` est le nom du script exécuté

Pour insérer un `message` dans le journal du script, ajouter la commande suivante dans le script :

```
echo "message"
```

14.3.2 Sortie dans le journal du module

Pour insérer des messages de niveau E ou I dans le journal du module, ajouter la commande suivante dans le script :

- en Windows

```
"%SAFE%/safekit" printe "message"
"%SAFE%/safekit" printi "message"
```

- en Linux

```
"$SAFE/safekit" printe "message"
"$SAFE/safekit" printi "message"
```

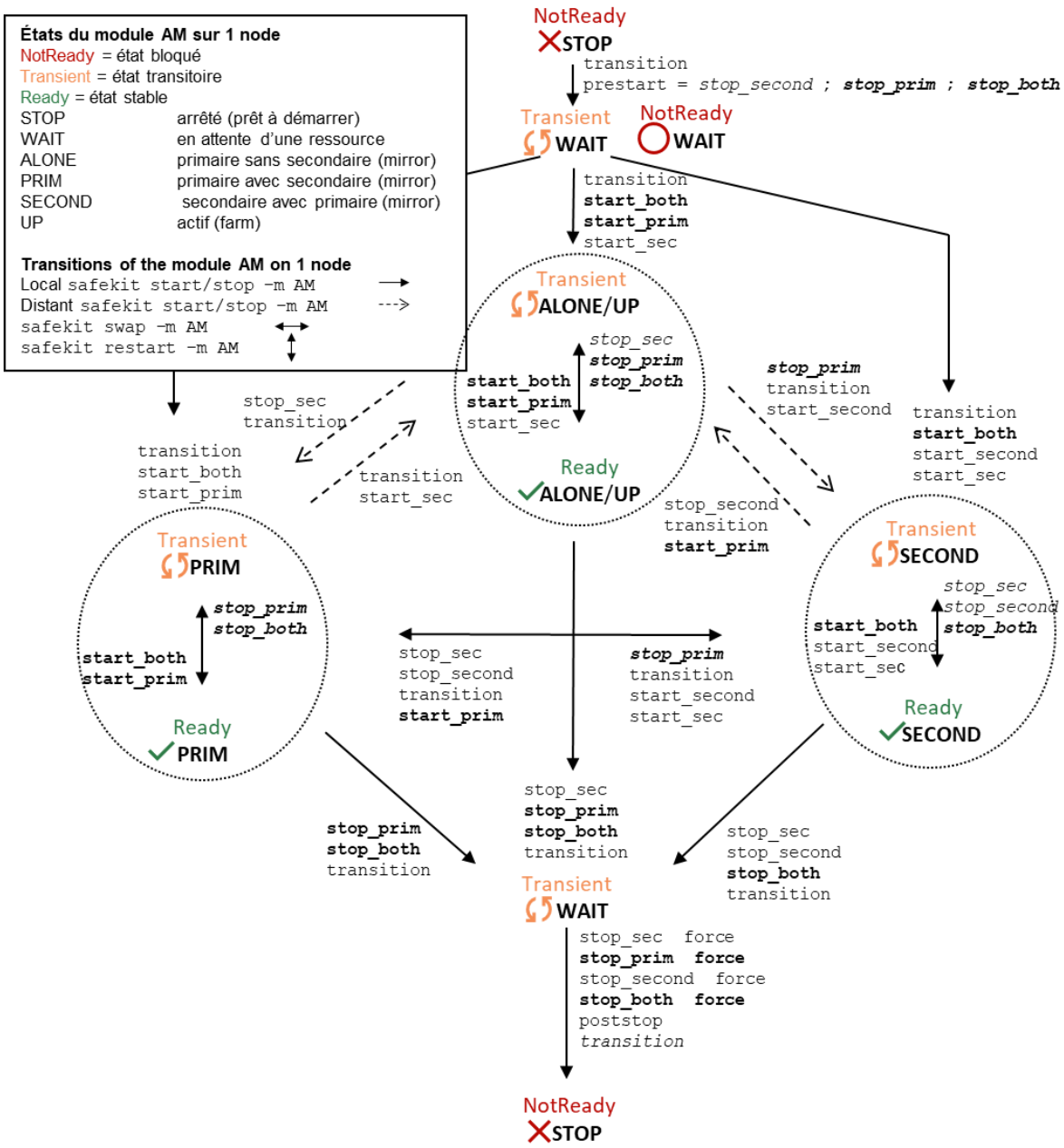
Les messages de niveau E sont visibles dans le journal non verbeux ; ceux de niveau I dans le journal verbeux.

Dans l'environnement d'exécution du script du module *AM*, l'option `-m AM` est inutile pour la commande `safekit`.

14.4 Automate d'exécution des scripts

La plupart du temps les scripts stop sont appelés 2 fois (sans l'option force puis avec l'option force). Dans ce cas, le nom du script est en italique.

Par exemple, au démarrage, la première transition de `STOP` vers `WAIT` appelle le script `transition STOP WAIT`.



14.5 Commandes spéciales SafeKit pour les scripts

Les commandes spéciales sont sous `SAFE/private/bin`. Les commandes peuvent être appelées directement dans les scripts du module avec :

- `%SAFEBIN%\specialcommand` en Windows
- `$SAFEBIN/specialcommand` en Linux

En dehors des scripts, il faut utiliser la commande `safekit -r`.

`safekit -r <special command> [<args>]`	<special command> <args> exécuté dans l'environnement SafeKit. Lorsque le path absolu n'est pas spécifié, la commande est recherchée dans `SAFEBIN=SAFE/private/bin`.

14.5.1 Commandes pour Windows

14.5.1.1 Commandes sleep, exitcode, sync

- `%SAFEBIN%\sleep.exe <timeout value in seconds>`

  A utiliser dans les scripts stop car net stop service n'est pas synchrone.

- `%SAFEBIN%\exitcode.exe <exit value>`

  Pour sortir d'un script avec une valeur d'erreur

- `%SAFEBIN%\sync.exe \\.\<drive letter:>`

  Pour synchroniser les caches file system

14.5.1.2 Commande namealias

`%SAFEBIN%/namealias [-n | -s ] <alias name>`

-n pour ajouter un nouveau nom NetBIOS en alias (`start_prim`) ou -s pour supprimer un alias NetBIOS (`stop_prim`)

Vous pouvez utiliser la commande SafeKit `netnames` (ou la commande Windows `nbtstat`) pour lister les informations NetBIOS.

14.5.2 Commandes pour Linux

14.5.2.1 Gestion de la crontab

`$SAFEBIN/gencron [del | add] <user name> [all |<command name>] -c "<comment>"`	• `del` pour désactiver des entrées dans `stop_prim` (en insérant des commentaires) • `add` pour activer des entrées dans `start_prim` (en retirant les commentaires). • *<user name>* nom de l'utilisateur dans la crontab • `all` pour s'appliquer à toutes les entrées • *<command name>* pour s'appliquer uniquement à au nom de la commande • *<comment>* entête du commentaire qui sera inséré

L'exemple suivant s'applique à l'entrée suivante dans crontab :

```
5 0 * * * $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
```

Pour un module miroir, pour gérer cette entrée sur le primaire, insérer :

- son activation dans `start_prim`

```
$SAFEBIN/gencron add admin daily.job -c "SafeKit configuration for
$SAFEMODULE"
```

- sa désactivation dans `stop_prim`

```
$SAFEBIN/gencron del admin daily.job -c "SafeKit configuration for
$SAFEMODULE"
```

14.5.2.2 Commande bounding

`$SAFEBIN/boundcmd <timeout value> <command path> [<args>]`	• `<timeout value>` temps maximum imparti pour exécuter la commande • `<command path>` chemin de la commande à exécuter • `<args>` arguments optionnels de la commande

`boundcmd` retourne l'exit code de la commande si la commande se termine et 2 sinon.

Par exemple, pour flusher des données sur disque avec un timeout de 30 secondes :

```
$SAFEBIN/boundcmd 30 /bin/sync 1>/dev/null 2>&1
```

14.5.3 Commandes pour Windows et Linux

`safekit -r processtree list | kill …`	Liste les processus en cours d'exécution sous forme d'arbre d'appel (excepté pour l'option `all`) et arrête les processus (option `kill`) • `safekit -r processtree list all` Liste tous les processus en cours d'exécution. • `safekit -r processtree list <process command name>` Liste les processus dont le nom de commande est celui spécifié. • `safekit -r processtree kill <process command name>` Liste et arrête les processus en cours d'exécution dont le nom de commande est celui spécifié. • `safekit -r processtree list | kill <process command name>| all <regular expression on the full command - path and arguments>` Pour la syntaxe des expressions régulières : • en Windows, voir class CatlRegExp • en Linux, voir man regex • Exemples en Windows

	`` safekit -r processtree kill notepad.exe ".*myfile.*" safekit -r processtree list all "mirror" ``` • Exemples en Linux ``` safekit -r processtree kill vi ".*myfile.*" safekit -r processtree list all "mirror" ```
`safekit incloop -m` *`AM`* `-i <handler name>`	Le module dispose d'un compteur `maxloop`, du nombre de `restart`, `stopstart` et `wait` du module sur détection d'erreur. Le module est arrêté lorsque ce compteur atteint la valeur `maxloop` sur la période `loop_interval`. Lors de l'exécution d'un handler spécial, le compteur `maxloop` n'est pas incrémenté. Pour l'incrémenter, utiliser la commande : `safekit incloop -m` *`AM`* `-i <handler name>` La commande augmente le compteur `maxloop` pour le module *`AM`* et retourne 1 lorsque la limite est atteinte. Pour un exemple, voir la section 15.4.2.
`safekit resetloop -m` *`AM`* `[-i <handler name>]`	Réinitialise le compteur `maxloop` pour le module *`AM`* (affectation de la valeur à 0)
`safekit checkloop -m` *`AM`*	Pour tester le compteur `maxloop` pour le module *`AM`*, utilisez la commande `safekit checkloop -m` *`AM`* • elle retourne 0 lorsque la limite `maxloop` n'est pas atteinte ou si la dernière incrémentation a eu lieu en dehors de la période `loop_interval` • elle retourne 1 quand la limite `maxloop` a été atteinte dans la période `loop_interval`

15.Exemples de configurations de module

Certains exemples décrits sont tirés des modules livrés avec le package SafeKit, sous `SAFE/Application_Modules`. De nombreux réels exemples d'intégration sont aussi décrits sous SafeKit Quick Installation Guides.

Les .safe sont plate-forme dépendants et, par conséquent, différents en Windows et Linux, principalement en ce qui concerne les scripts du module.

La configuration du module peut être modifiée :

- soit via l'assistant de configuration du module de la console web SafeKit (voir la section 3.3)
- soit en éditant directement les fichiers `SAFE/module/`*`AM`*`/conf/userconfig.xml` ou les scripts sous `SAFE/module/`*`AM`*`/bin` (où *`AM`* est le nom du module installé)

Pour être prise en compte, au prochain démarrage du module, la nouvelle configuration doit être appliquée :

- soit à la dernière étape de l'assistant de configuration du module
- soit avec la commande `safekit config -H "`*`node1`*`,`*`node2`*`" -E `*`AM`* exécutée sur le nœud sur lequel les fichiers ont été modifiés

Avant d'appliquer la configuration, fermez tous les éditeurs, explorateur de fichiers, shells ou cmd qui accèderaient un fichier sous `SAFE/modules/`*`AM`* sur les nœuds.

15.1 Exemple de module miroir avec `mirror.safe`

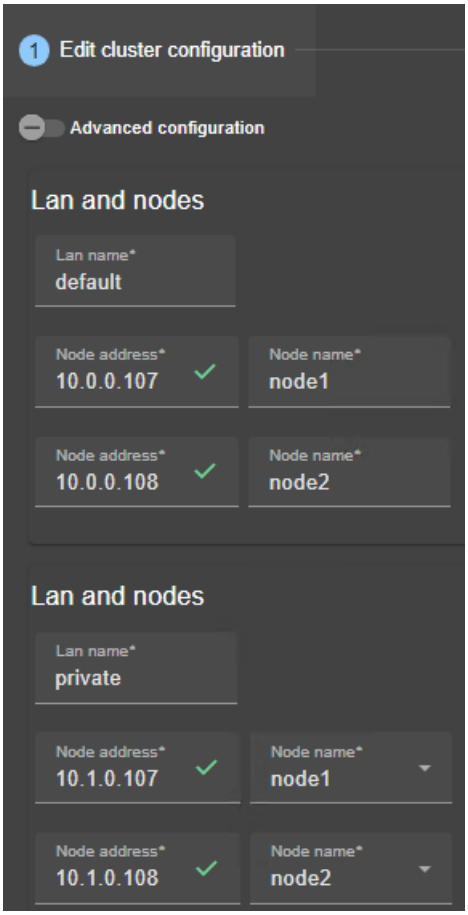
Ci-dessous le fichier de configuration et les scripts du module du module miroir générique, `mirror.safe`, pour Windows.

Pour tester un module miroir, voir la section 4.2.

La description suivante est pour Windows. Pour Linux, veuillez consulter `mirror.safe` fourni avec le package Linux qui comprend la configuration et les scripts pour Linux.

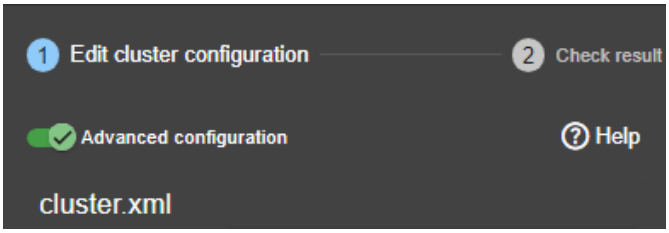
15.1.1 Configuration du cluster avec deux réseaux

La configuration du cluster comprend deux réseaux, nommés par exemple `default` et `private`. Le deuxième réseau est défini pour illustrer la configuration d'un réseau dédié au trafic de réplication dans la configuration du module. La plupart des configurations incluent généralement un seul réseau.



Pour lancer l'assistant de configuration du cluster, voir section 3.2.

Basculer en « Configuration avancée » pour éditer le XML si besoin.




```
<cluster>
   <lans>
      <lan name="default">
         <node name="node1"
addr="10.0.0.107"/>
         <node name="node2"
addr="10.0.0.108"/>
      </lan>
      <lan name="private">
         <node name="node1"
addr="10.1.0.107"/>
         <node name="node2"
addr="10.1.0.108"/>
      </lan>
   </lans>
</cluster>
```

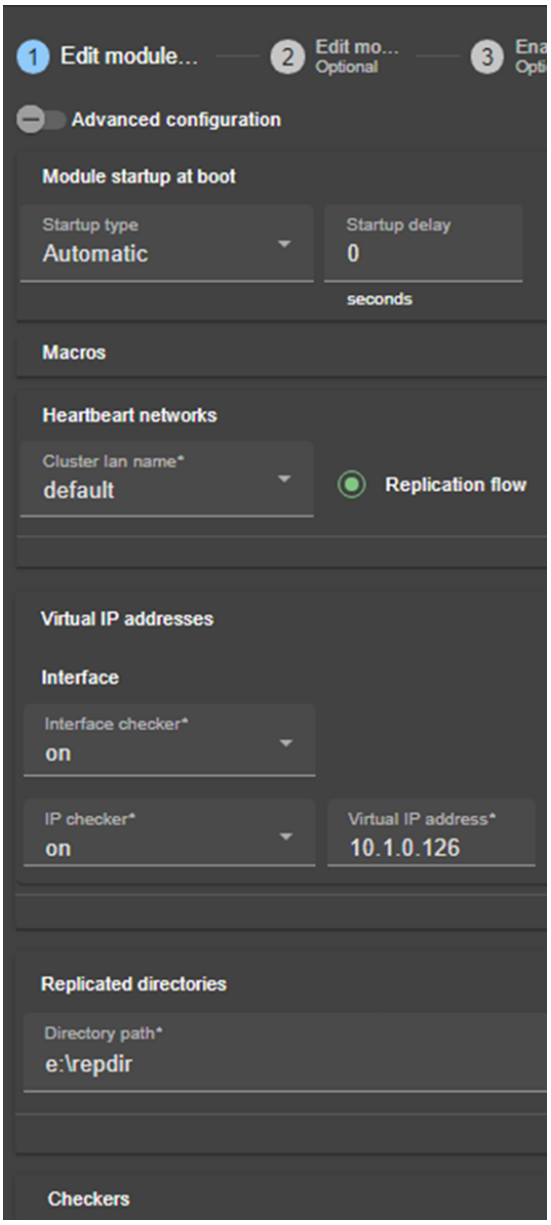
Pour des détails sur la configuration XML, voir section 12.1.

15.1.2 Configurations du module miroir

Ci-dessous des configurations d'un module miroir avec une adresse IP virtuelle, la réplication de fichiers en temps réel et le basculement automatique.

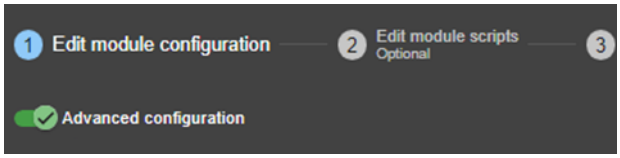
15.1.2.1 Configuration avec une adresse IP virtuelle, la réplication de fichiers en temps réel et le basculement automatique

La configuration suivante utilise seulement un seul réseau pour le trafic des heartbeats et de la réplication.



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.



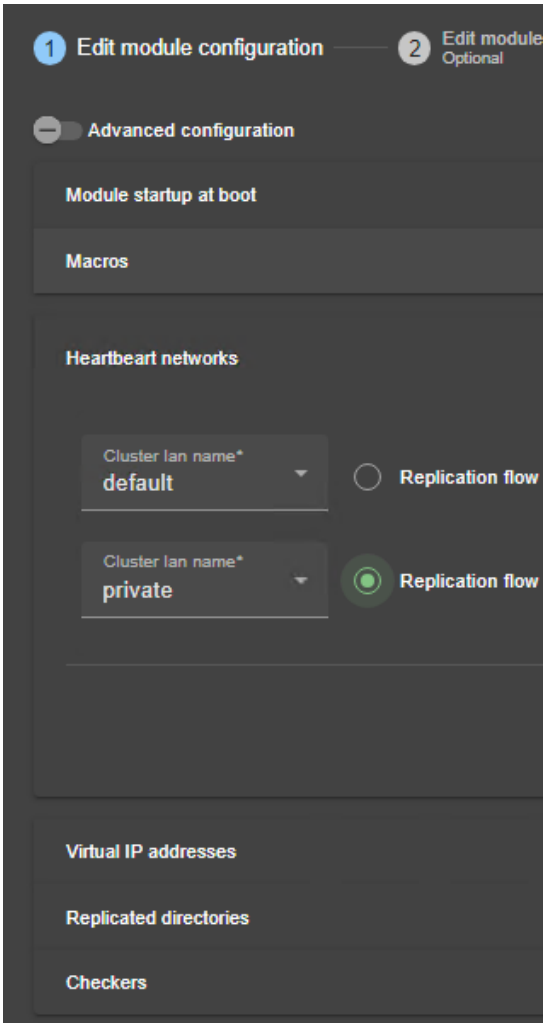

```
<!DOCTYPE safe>
<safe>
<service mode="mirror" boot="on">

 <heart>
  <heartbeat name="default">
  </heartbeat>
 </heart>

 <vip>
  <interface_list>
   <interface check="on">
    <real_interface>
     <virtual_addr addr="10.1.0.126"
where="one_side_alias" check="on"/>
    </real_interface>
   </interface>
  </interface_list>
 </vip>

 <rfs>
  <replicated dir="e:\repdir"/>
 </rfs>

 <user></user>

</service>
   </safe>
```

Pour des détails sur la configuration XML :

- `<service>` voir section 13.2
- `<heart>` voir section 13.3
- `<vip>` voir section 13.5
- `<rfs>` voir section 13.6
- `<user>` voir section 13.7

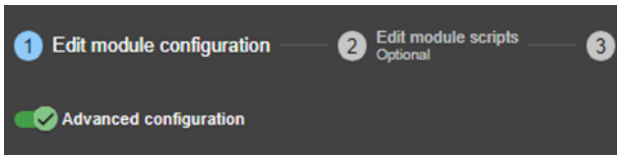
15.1.2.2 Configuration avec un réseau de réplication dédié

Le module est configuré pour utiliser les deux réseaux du cluster définis en section 15.1.1. Celui nommé `private` est sélectionné comme « Flux de réplication ».



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.




```
<!DOCTYPE safe>
<safe>
<service mode="mirror" boot="on">

 <heart>
  <heartbeat name="default"/>
  <heartbeat name="private"
ident="flow"/>
 </heart>

 <vip>
  <interface_list>
   <interface check="on">
    <real_interface>
     <virtual_addr addr="10.1.0.126"
where="one_side_alias" check="on"/>
    </real_interface>
   </interface>
  </interface_list>
 </vip>

 <rfs>
  <replicated dir="e:\repdir"/>
 </rfs>

 <user></user>

</service>
</safe>
```

Pour la description de la configuration XML voir section 13.

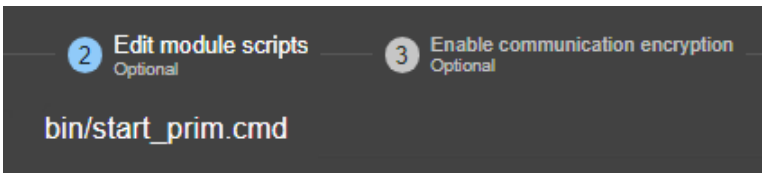
15.1.3 Scripts du module miroir

Ci-dessous les scripts du module, en CMD sur Windows, pour démarrer/arrêter les service du module miroir sur le nœud primaire. Pour Linux, se référer au module `mirror.safe` livré avec le package Linux.

Pour plus de détails sur les scripts du module, voir section 14.
Pour plus de détails sur la sortie des scripts (avec `echo` et `safekit printe`), voir la section 14.3.

15.1.3.1 Script start_prim

Le script `start_prim` est appelé lorsque le module miroir démarre en tant que primaire (démarrage manuel ou automatique après un `stopstart` ou sortie du `wait`) ou redémarre sur un nœud primaire (`restart`). Il doit contenir le démarrage de l'application intégrée dans le module. L'application s'exécute uniquement sur le nœud primaire.

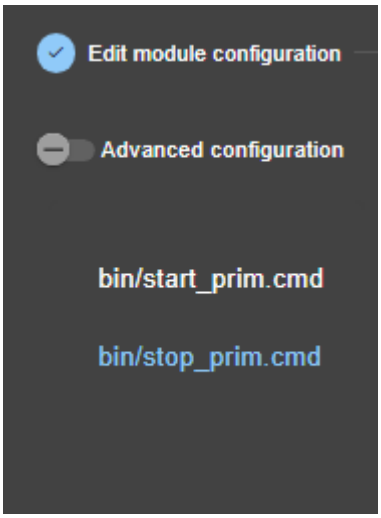


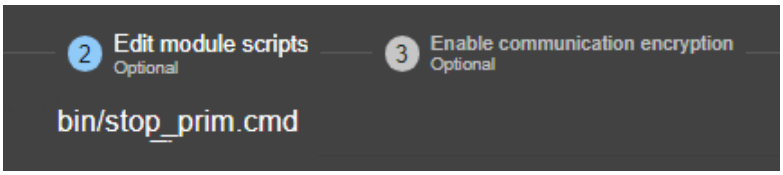

```
@echo off
echo "Running start_prim %*"
set res=0

rem net start "myservice"

set res=%errorlevel%
if %res% == 0 goto end

:stop
"%SAFE%\safekit" printe "start_prim failed"

rem uncomment to stop the module when critical
rem "%SAFE%\safekit" stop -i "start_prim"

:end
```

Pour la description de la commande `safekit`, voir la section 9.

15.1.3.2 Script stop_prim

Le script `stop_prim` est appelé lorsque le module s'arrête (`stop`, `stopstart` ou `wait`) ou redémarre sur un nœud primaire (`restart`). Il doit contenir l'arrêt de l'application intégrée dans le module






```
@echo off
echo "Running stop_prim %*"
set res=0

rem default: no action on forcestop
if "%1" == "force" goto end

rem net stop "myservice" /Y
set res=%errorlevel%

rem If necessary, wait for the stop of the services
rem "%SAFEBIN%\sleep" 10
```

```
if %res% == 0 goto end

"%SAFE%\safekit" printe "stop_prim failed"

:end
```

15.2 Exemple de module ferme avec `farm.safe`

Ci-dessous le fichier de configuration et les scripts du module pour le module ferme générique, `farm.safe`, pour Windows.

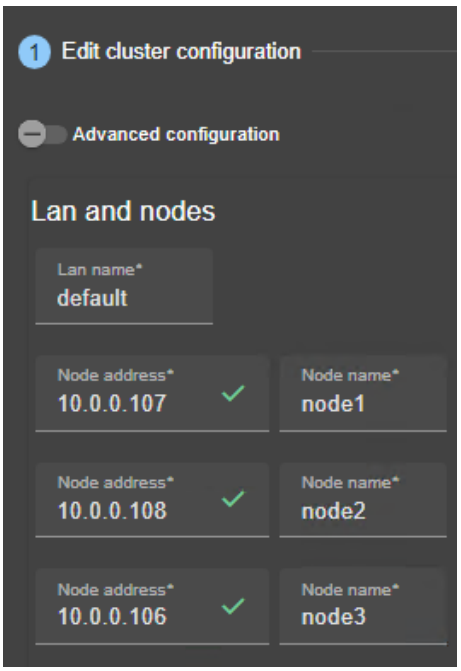
Pour tester un module ferme, voir la section 4.3.

La description suivante est pour Windows. Pour Linux, veuillez consulter `farm.safe` fourni avec le package Linux qui comprend la configuration et les scripts pour Linux.

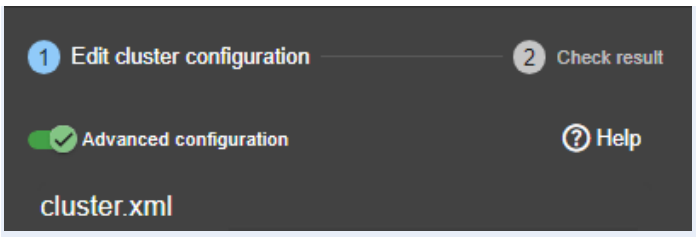
15.2.1 Configuration du cluster avec trois nœuds

La configuration du cluster comprend un seul réseau nommé `default` et trois nœuds pour démontrer une configuration avancée de répartition de charge. La plupart des configurations de cluster incluent seulement deux nœuds.

Seul un module de ferme avec équilibrage de charge et sans réplication peut être configuré sur plus de deux nœuds. Un module miroir avec réplication ne peut être configuré que sur deux nœuds.



Pour lancer l'assistant de configuration du cluster, voir section 3.2.




```
<cluster>
   <lans>
      <lan name="default">
         <node name="node1"
addr="10.0.0.107"/>
         <node name="node2"
addr="10.0.0.108"/>
         <node name="node3"
addr="10.0.0.106"/>
      </lan>

   </lans>
</cluster>
```

Pour des détails sur la configuration XML, voir section 12.1.

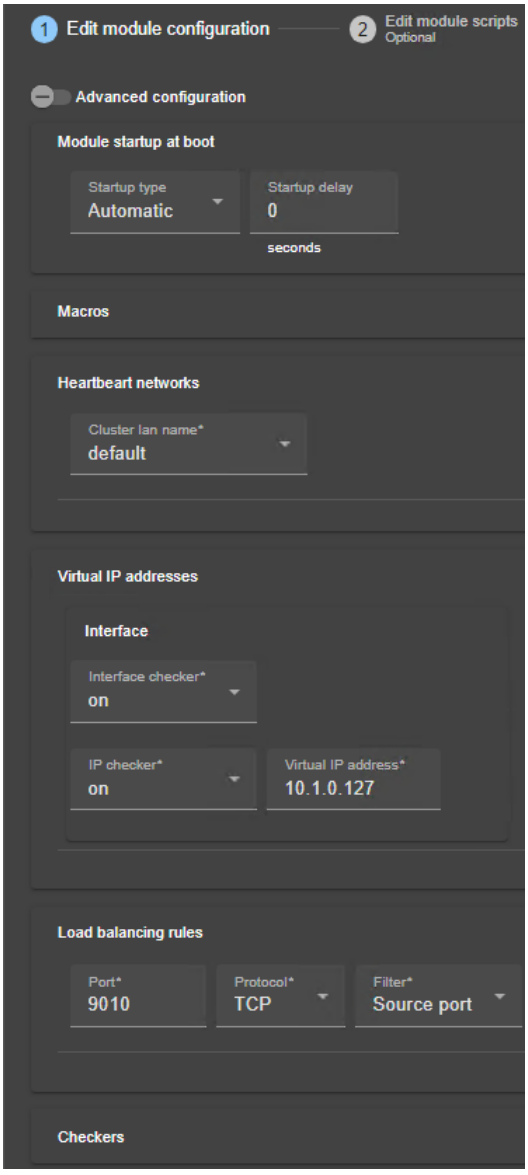
Basculer en « Configuration avancée » pour éditer le XML si besoin	

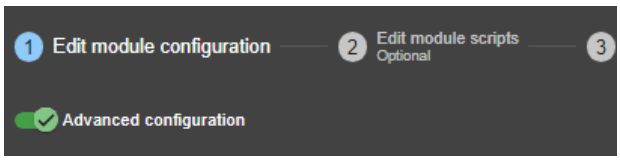
15.2.2 Configurations du module ferme

Ci-dessous des configurations d'un module ferme avec une adresse IP virtuelle, des règles d'équilibrage de charge et le basculement automatique.

15.2.2.1 Configuration avec une adresse IP virtuelle, l'équilibrage de charge et le basculement automatique

La règle de répartition de charge définie ci-dessous permet de visualiser la distribution de la charge entre les nœuds en accédant à la page web de SafeKit à l'adresse http://host:9010/safekit/mosaic.html.






```
<!DOCTYPE safe>
<safe>
 <service mode="farm" boot="on">
  <farm>
   <lan name="default"></lan>
  </farm>
  <vip>
   <interface_list>
    <interface check="on">
     <virtual_interface
type="vmac_directed">
      <virtual_addr addr="10.1.0.127"
where="alias" check="on"/>
     </virtual_interface>
    </interface>
   </interface_list>

   <loadbalancing_list>
    <group name="FarmProto">
     <rule port="9010" proto="tcp"
filter="on_port"/>
    </group>
   </loadbalancing_list>
  </vip>

  <user></user>

 </service>
</safe>
```

Pour des détails sur la configuration XML de :

- `<service>` voir section 13.2
- `<farm>` voir section 13.4

Pour lancer l'assistant de configuration du module, voir section 3.3. Basculer en « Configuration avancée » pour éditer le XML si besoin.	• `<vip>` voir section 13.5 • `<loadbalancing_list>` voir section 13.5.6 • `<user>` voir section 13.7

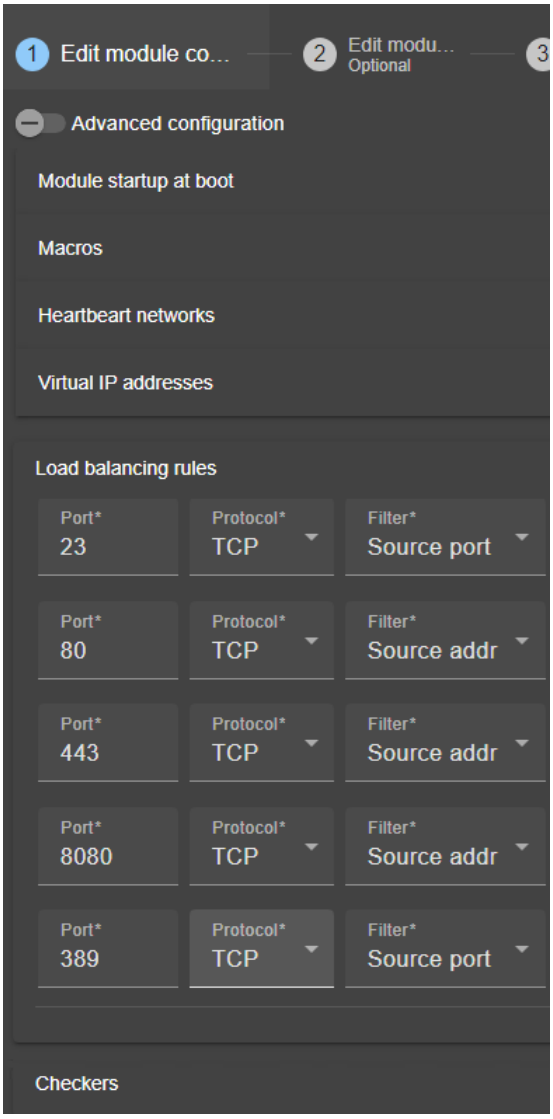
D'autres exemples de configuration des règles d'équilibrage de charge sont décrits ci-dessous.

15.2.2.2 Configuration avec des règles d'équilibrage de charge pour TCP

Voici la configuration de l'équilibrage de charge pour accéder à l'adresse IP virtuelle en utilisant le protocole TCP sur les ports spécifiés :

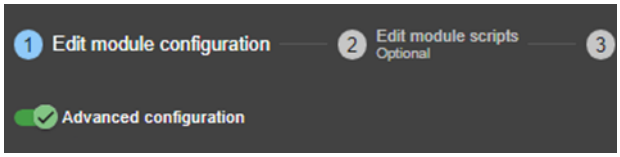
- 80 (HTTP), 443 (HTTPS), 8080 (HTTP proxy)

  Avec HTTP et HTTPS, le partage de charge est défini sur l'adresse IP client (`"on_addr"`) et non sur le port TCP client (`"on_port"`), pour assurer que le même client est toujours connecté sur le même serveur sur plusieurs connexions TCP (service à état versus service sans état ; voir la description en section 1.3.3).
- 389 (LDAP) et 23 (Telnet)



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.




```
<!DOCTYPE safe>
<safe>
 <service mode="farm" boot="on">
  <farm>
    <lan name="default"></lan>
  </farm>
  <vip>
   <interface_list>
    <interface check="on">
     <virtual_interface
type="vmac_directed">
      <virtual_addr addr="10.1.0.127"
where="alias" check="on"/>
     </virtual_interface>
    </interface>
   </interface_list>
   <loadbalancing_list>
    <group name="FarmProto">
     <rule port="23" proto="tcp"
filter="on_port"/>
     <rule port="80" proto="tcp"
filter="on_addr"/>
     <rule port="443" proto="tcp"
filter="on_addr"/>
     <rule port="8080" proto="tcp"
filter="on_addr"/>
     <rule port="389" proto="tcp"
filter="on_port"/>
    </group>
   </loadbalancing_list>
  </vip>
  <user></user>
 </service>
</safe>
```

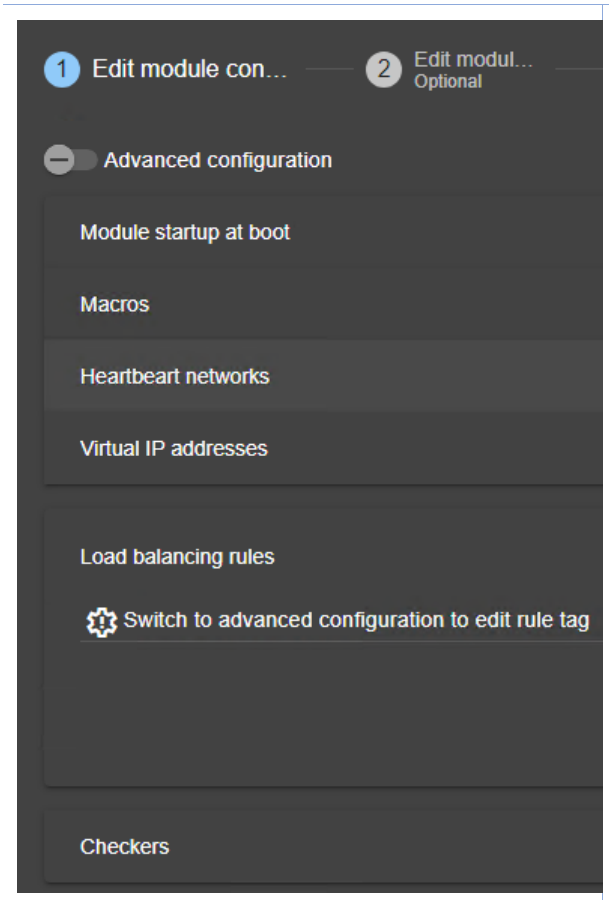
Pour une description de `<loadbalancing_list>` voir section 13.5.6.

15.2.2.3 Configuration avec des règles d'équilibrage de charge pour UDP

Voici la configuration de l'équilibrage de charge pour accéder à l'adresse IP virtuelle en utilisant le protocole UDP sur les ports spécifiés :

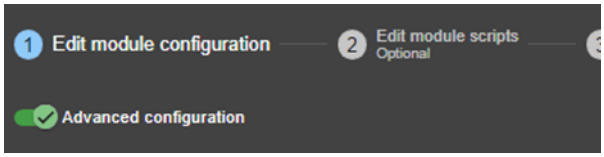
- 53 (DNS)
- 1645 (RADIUS)

Avec `on_ipid`, l'équilibrage de charge est réalisé sur le champ "IP identifier" dans l'entête IP du paquet. L'équilibrage fonctionne même si le client présente la même adresse IP client et le même port en entrée.



Pour lancer l'assistant de configuration du module, voir section 3.3.

L'assistant ne propose pas le filtre `on_ipid`. Basculer en « Configuration avancée » pour l'éditer.

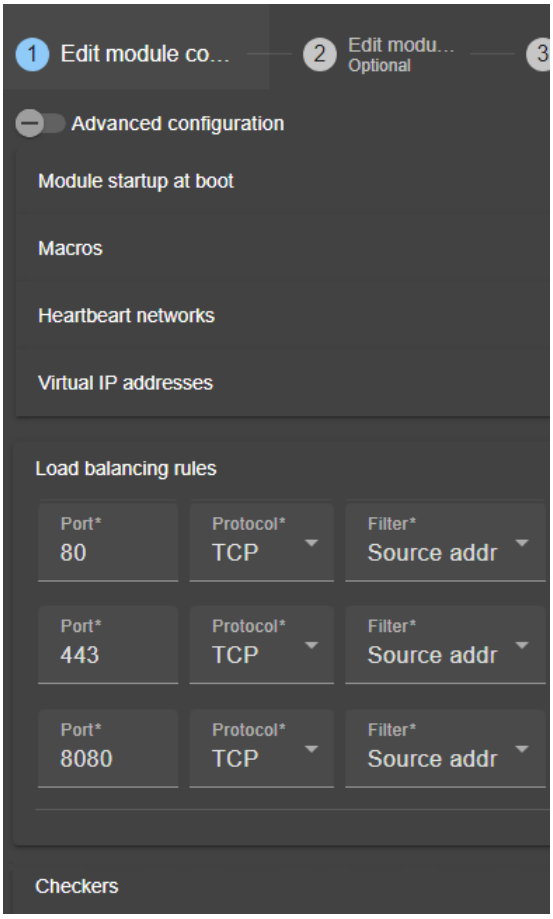



```
<!DOCTYPE safe>
<safe>
 <service mode="farm" boot="on">
  <farm>
   <lan name="default"></lan>
  </farm>
  <vip>
   <interface_list>
    <interface check="on">
     <virtual_interface
type="vmac_directed">
      <virtual_addr addr="10.1.0.127"
where="alias" check="on"/>
     </virtual_interface>
    </interface>
   </interface_list>
   <loadbalancing_list>
    <group name="FarmProto">
     <rule port="53" proto="udp"
filter="on_ipid" />
     <rule port="1645"  proto="udp"
filter="on_ipid" />
    </group>
   </loadbalancing_list>
  </vip>

  <user></user>
 </service>
</safe>
```

Pour une description de `<loadbalancing_list>` voir section 13.5.6.

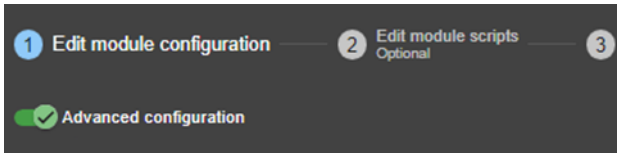
15.2.2.4 Configuration avancée de l'équilibrage de charge

Avec la configuration suivante, vous définissez une ferme de 3 nœuds avec une priorité pour le trafic HTTP pour node1, HTTPS pour node2 et proxy HTTP pour le node3.



Pour lancer l'assistant de configuration du module, voir section 3.3.

L'assistant n'affiche pas les détails sur les groupes d'équilibrage de charge. Basculer en « Configuration avancée » pour les éditer.




```
<!DOCTYPE safe>
<safe>
 <service mode="farm" boot="on">
  <farm>
    <lan name="default"></lan>
  </farm>
  <vip>
   <interface_list>
    <interface check="on">
     <virtual_interface
type="vmac_directed">
      <virtual_addr addr="10.1.0.127"
where="alias" check="on"/>
     </virtual_interface>
    </interface>
   </interface_list>
   <loadbalancing_list>
     <group name="http_service">
       <cluster>
         <host name="node1" power="3"/>
         <host name="node2" power="1"/>
         <host name="node3" power="1"/>
       </cluster>
       <rule port="80" proto="tcp"
filter="on_addr"/>
      </group>
      <group name="https_service">
        <cluster>
         <host name="node1" power="1"/>
         <host name="node2" power="3"/>
         <host name="node3" power="1"/>
        </cluster>
        <rule port="443" proto="tcp"
filter="on_addr"/>
      </group>
      <group name="httpproxy_service">
       <cluster>
        <host name="node1" power="1"/>
        <host name="node2" power="1"/>
        <host name="node3" power="3"/>
       </cluster>
       <rule port="8080" proto="tcp"
filter="on_addr"/>
      </group>
   </loadbalancing_list>
  </vip>
  <user></user>
 </service>
</safe>
```

	Pour une description de `<loadbalancing_list>` voir section 13.5.6.

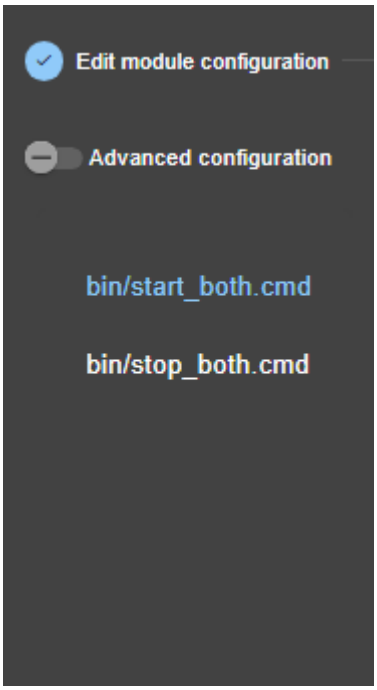
15.2.3 Scripts du module ferme

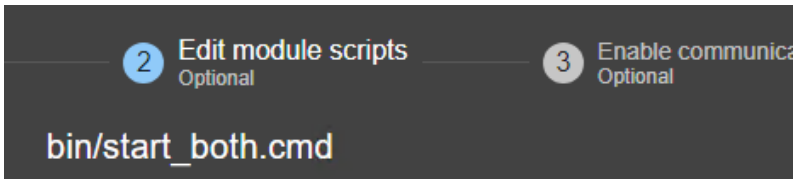
Ci-dessous les scripts du module, en CMD sur Windows, pour démarrer/arrêter les service du module ferme sur tous les nœuds. Pour Linux, se référer au module `farm.safe` livré avec le package Linux.

Pour plus de détails sur les scripts du module, voir section 14.
Pour plus de détails sur la sortie des scripts (avec `echo` et `safekit printe`), voir la section 14.3.

15.2.3.1 Script start_both

Le script `start_both` est appelé lorsque le module de ferme démarre (démarrage manuel ou automatique après un `stopstart` ou sortie du `wait`) ou redémarre (`restart`). Il doit contenir le démarrage de l'application intégrée dans le module. L'application s'exécute sur tous les nœuds.



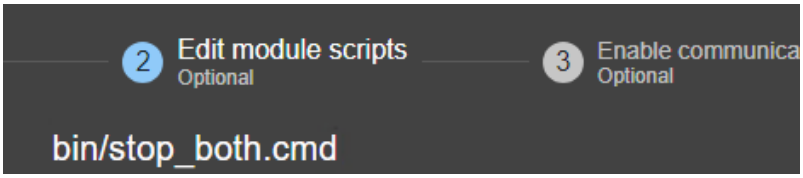



```
@echo off
echo "Running start_both %*"
set res=0

rem net start "myservice"

set res=%errorlevel%
if %res% == 0 goto end

:stop
"%SAFE%\safekit" printe "start_both failed"

rem uncomment to stop the module when critical
rem "%SAFE%\safekit" stop -i "start_both"

:end
```

Pour la description de la commande `safekit`, voir la section 9.

15.2.3.2 Script stop_both

Le script `stop_both` est appelé lorsque le module de ferme s'arrête (`stop`, `stopstart` ou `wait`) ou redémarre (`restart`). Il doit contenir l'arrêt de l'application intégrée dans le module.




```
@echo off
echo "Running stop_both %*"
set res=0

rem default: no action on forcestop
if "%1" == "force" goto end

rem net stop "myservice" /Y
set res=%errorlevel%

rem If necessary, wait for the stop of the services
rem "%SAFEBIN%\sleep" 10

if %res% == 0 goto end

"%SAFE%\safekit" printe "stop_both failed"

:end
```

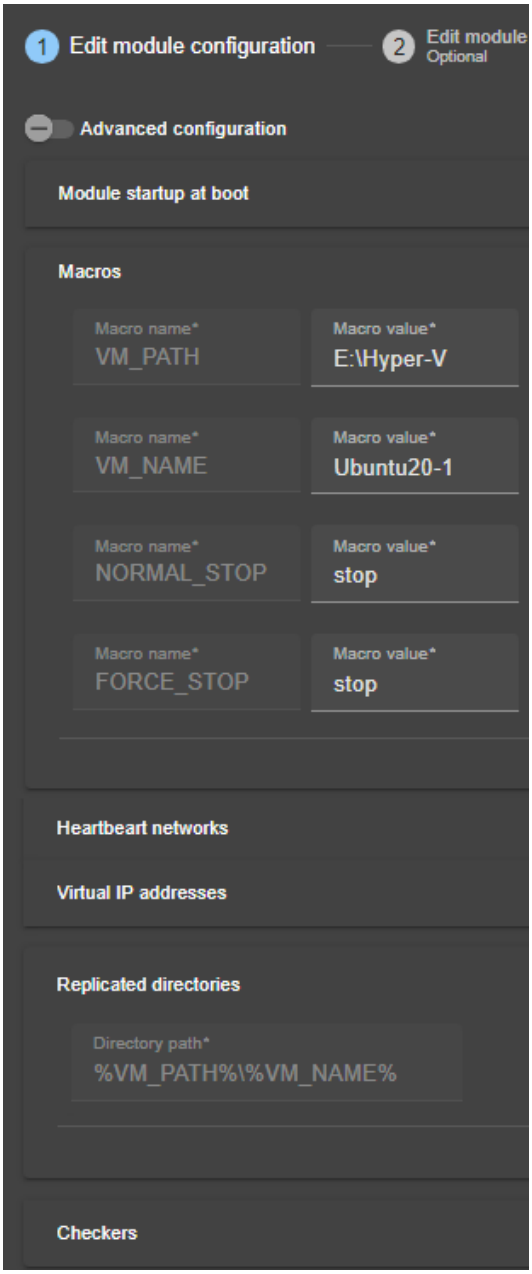
Pour la description de la commande `safekit`, voir la section 9.

15.3 Exemple d'utilisation de macros et variables de script avec `hyperv.safe`

Le module `hyperv.safe` apporte la haute disponibilité à Hyper-V pour deux serveurs Windows. Il s'agit d'un module miroir, avec une adresse IP virtuelle, la réplication temps réel et le basculement automatique. Il est présenté pour démontrer l'utilisation de macros et de variables d'environnement des scripts du module.

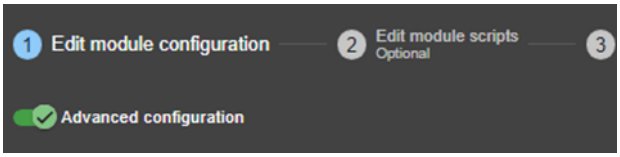
15.3.1 Configuration du module avec macro et var

Dans l'exemple suivant, quatre `<macro>` sont configurés, et leurs valeurs sont utilisées pour définir le chemin du répertoire répliqué `<dir>` (c'est-à-dire `E:\Hyper-V\Ubuntu20-1`) et les variables d'environnement `<var>` pour les scripts de module. Noter que dans l'exemple, le nom des macros et des variables sont identiques, ce qui n'est pas une obligation.



Pour lancer l'assistant de configuration du module, voir section 3.3.

L'assistant n'affiche pas la section `user`. Basculer en « Configuration avancée » pour l'éditer.




```
<!DOCTYPE safe>
<safe>
 <macro name="VM_PATH"
        value="E:\Hyper-V"/>
 <macro name="VM_NAME"
        value="Ubuntu20-1"/>
 <macro name="NORMAL_STOP"
        value="stop"/>
 <macro name="FORCE_STOP"
        value="stop"/>
 <service mode="mirror">
  <heart>
    <heartbeat name="default" />
  </heart>
  <rfs scripts="on" acl="on"
namespacepolicy="0" allocthreshold="10">
    <replicated
dir="%VM_PATH%\%VM_NAME%">
    </replicated>
  </rfs>
  <user>
    <var name="VM_PATH"
         value="%VM_PATH%\%VM_NAME%"/>
    <var name="VM_NAME"
         value="%VM_NAME%"/>
    <var name="NORMAL_STOP"
         value="%NORMAL_STOP%"/>
    <var name="FORCE_STOP"
         value="%FORCE_STOP%"/>
   </user>
 </service>
</safe>
```

Pour des détails sur la configuration XML de :

- `<macro>` voir section 13.1
- `<user>`, `<var>` voir section 13.7

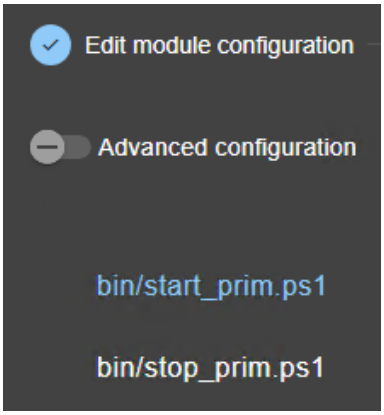
15.3.2 Accès des variables par les scripts du module

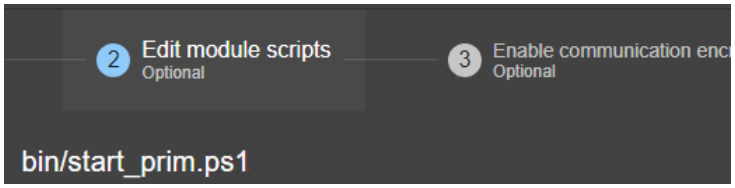
Ci-dessous, le script `start_prim.ps1` accèdent aux variables d'environnement définies lors de son exécution :

- Les variables d'environnement SafeKit `SAFE` et `SAFEUSERVAR`

- Les variables d'environnement du module `VM_PATH`, `VM_NAME`… définies dans les sections `<var>` de `<user>`

Pour plus de détails, voir section 14.2.






```
$safe = (Get-Item env:SAFE).Value
$suv = (Get-Item env:SAFEUSERVAR).Value
$vmname = (Get-Item env:VM_NAME).Value
$vmpath = (Get-Item env:VM_PATH).Value

…
```

15.4 Exemple de surveillance de processus avec `softerrd.safe`

Le module `softerrd.safe` est un module miroir de démonstration de la surveillance de processus. Cette fonctionnalité est disponible également dans un module ferme.

Les tests consistent à tuer les processus surveillés (i.e. `mybin`, `myotherbin` ou `myappli`) avec la commande `safekit kill`.

Pour tester la surveillance de processus, voir la section 4.4.1.

La description suivante est pour Windows. Pour Linux, veuillez consulter `softerrd.safe` fourni avec le package Linux qui comprend la configuration et les scripts pour Linux.

15.4.1 Configuration du module avec surveillance de processus

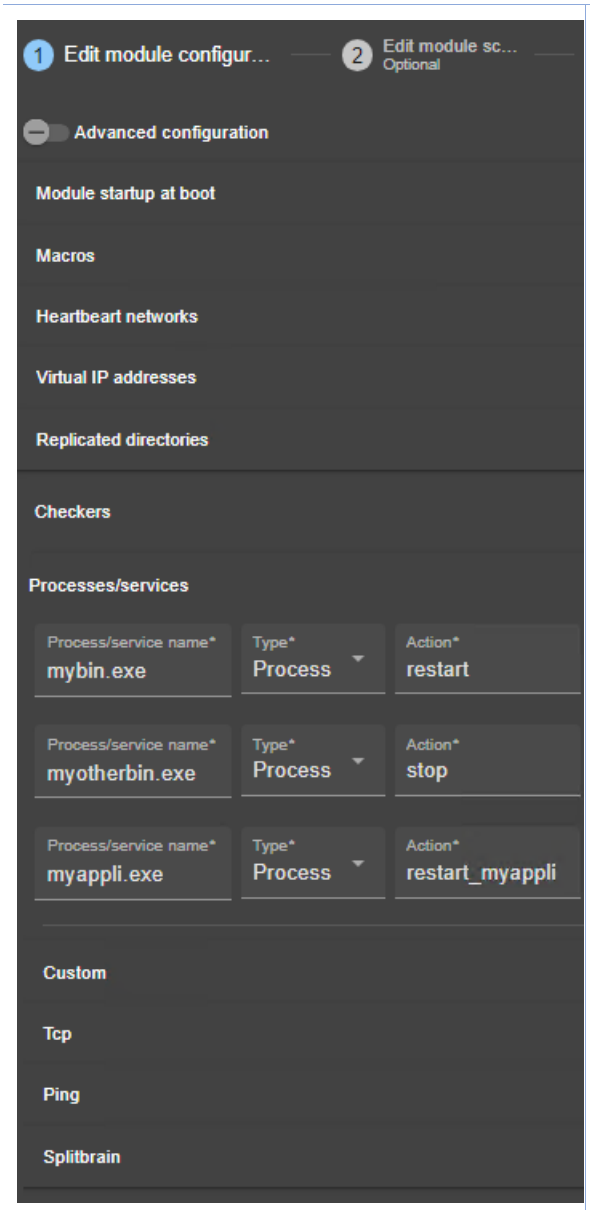
La détection de l'arrêt de :

- `mybin` provoque un restart du module (`action="restart"`). Sa surveillance est activée/désactivée après/avant l'exécution de `start_prim`/`stop_prim` (`class="prim"`).
- `myotherbin` provoque un stop du module (`action="stop"`). Sa surveillance est activée/désactivée après/avant l'exécution de `start_second`/`stop_ second` (`class="second"`).
- `myappli` provoque l'exécution d'un script spécial `restart_myappli.cmd` (`action="restart_myappli"`) localisé dans le répertoire `bin` du module. La surveillance de `myappli` est activée/désactivée manuellement, dans les scripts du module, après/avant son lancement/arrêt (e.g. `class="myappli"`). Voir les scripts dans la section 15.4.2.

  Cette configuration permet de redémarrer individuellement le processus `myappli` sans redémarrer complètement l'application intégrée dans `start_prim/stop_prim`.

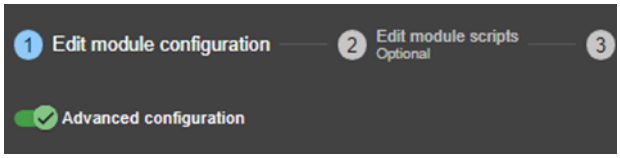
Les actions `restart` et `stopstart` incrémentent automatiquement le compteur `maxloop` pour limiter le nombre d'actions automatiques en cas d'erreur persistante. Par défaut, à la 4ème détection d'erreur en 24h (voir `maxloop` et `loop_interval` décrits dans la section 13.2.3), le module est arrêté.

Si l'action consiste en l'exécution d'un script spécial, ce script doit gérer manuellement le compteur `maxloop` (i.e. `restart_myappli.cmd`).



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.



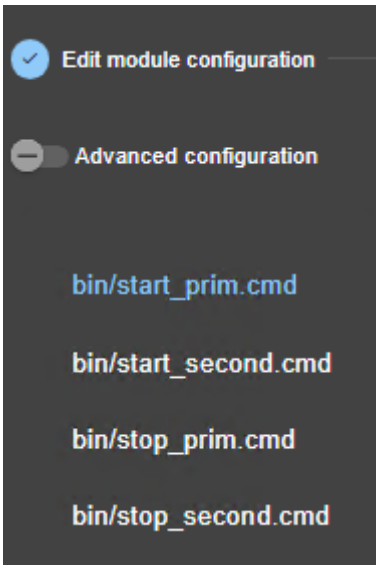

```
<!DOCTYPE safe>
<safe>
 <service mode="mirror">
   <heart>
     <heartbeat name="default">
     </heartbeat>
   </heart>

   <errd>
    <proc name="mybin.exe"
          action="restart"
          class="prim"/>
    <proc name="myotherbin.exe"
          action="stop"
          class="second"/>
    <proc name="myappli.exe"
          action="restart_myappli"
          class="myappli"/>
   </errd>

   <user>
   </user>

 </service>
</safe>
```

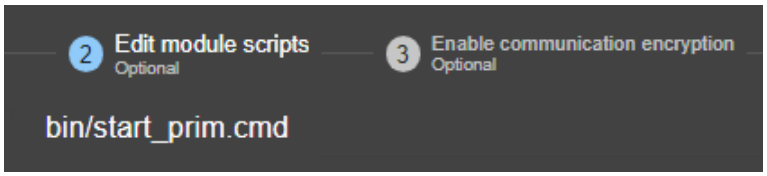
Pour des détails sur la configuration XML de `<errd>`, voir section 13.9.

15.4.2 Configuration avancée des scripts du module

Le module surveille la présence des processus :

- `mybin` et `myappli` démarrés/arrêtés sur le nœud primaire avec `start_prim`/`stop_prim`
- `myotherbin` démarré/arrêté sur le nœud secondaire avec `start_second`/`stop_second`

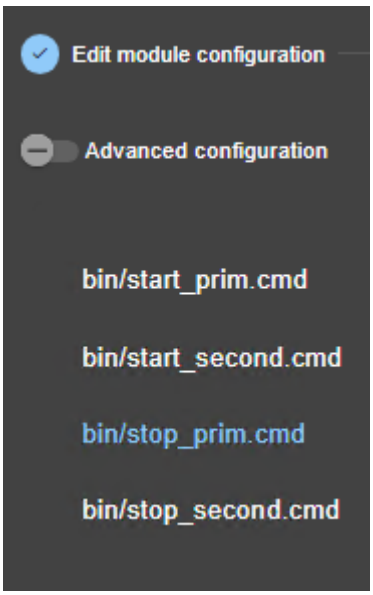


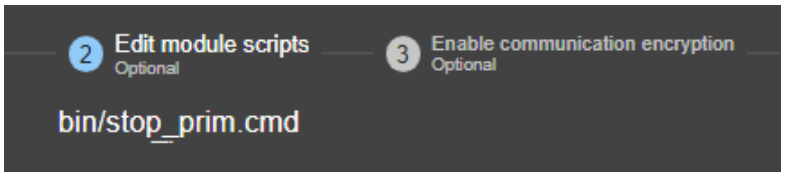



```
@echo off
echo "Running start_prim %*"

%SAFE%\safekit printi "start mybin"
start %SAFEUSERBIN%\mybin.exe 10000000

%SAFE%\safekit printi "start myappli"
start %SAFEUSERBIN%\myappli.exe 10000000
%SAFE%\safekit errd enable myappli
```

Noter l'appel à `safekit errd enable myappli` pour démarrer la surveillance du processus avec `class="myappli"`, une fois ce processus lancé.

Voir la section 13.9.4 pour la description de cette commande.

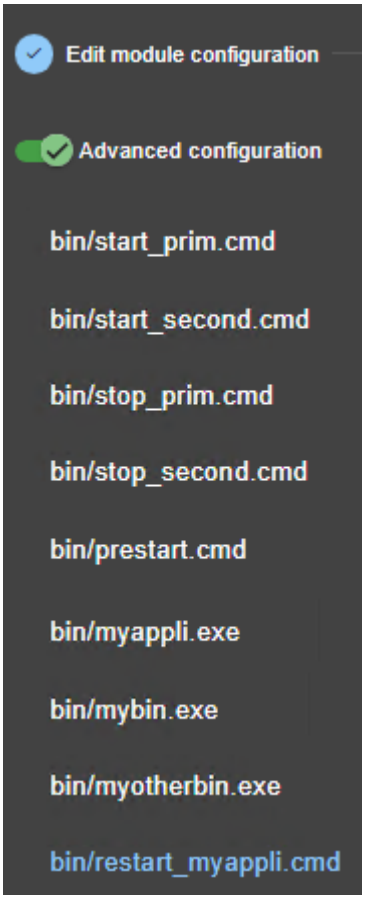




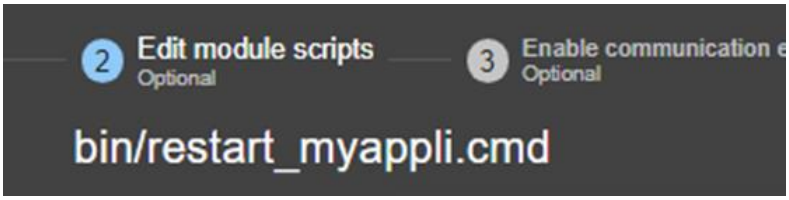

```
@echo off
echo "Running stop_prim %*"
%SAFE%\safekit printi "stop mybin"
%SAFE%\safekit kill -level="terminate" -
name="mybin.exe"

%SAFE%\safekit printi "stop myappli"
%SAFE%\safekit errd disable myappli
%SAFE%\safekit kill -level="terminate" -
name="restart_myappli.cmd"
%SAFE%\safekit kill -level="terminate" -
name="myappli.exe"
    :end
```

Noter l'appel à `safekit errd disable myappli` pour arrêter la surveillance du processus avec `class="myappli"`, avant d'arrêter ce processus.

Le handler spécifique `restart_myappli.cmd` est éditable en « Configuration avancée ». Ce script incrémente le compteur `maxloop` et redémarre le processus `myappli`.



Basculer en « Configuration avancée » pour lister et éditer tous les scripts.




```
@echo off

%SAFE%\safekit printi "restart_myappli"

rem first disable monitoring of the application
%SAFE%\safekit errd disable myappli

rem increment loop counter
%SAFE%\safekit incloop -i "restart_myappli"
if  %errorlevel% == 0 goto next
rem max loop reached
%SAFE%\safekit stop -i "restart_myappli"
%SAFEBIN%\exitcode 0

:next
rem max loop not reached : go on restarting the
application
%SAFE%\safekit printi "Restart myappli"
%SAFE%\safekit kill -level="terminate" -
name="myappli.exe"
start %SAFEUSERBIN%\myappli.exe 10000000

rem finally, enable monitoring of the application
%SAFE%\safekit errd enable myappli
%SAFEBIN%\exitcode 0
```

Noter l'incrémentation du compteur de rebouclage avec `safekit incloop -i "restart_myappli"` et l'arrêt du module quand `maxloop` est atteint.

Voir section 14.5.3 pour la description de cette commande.

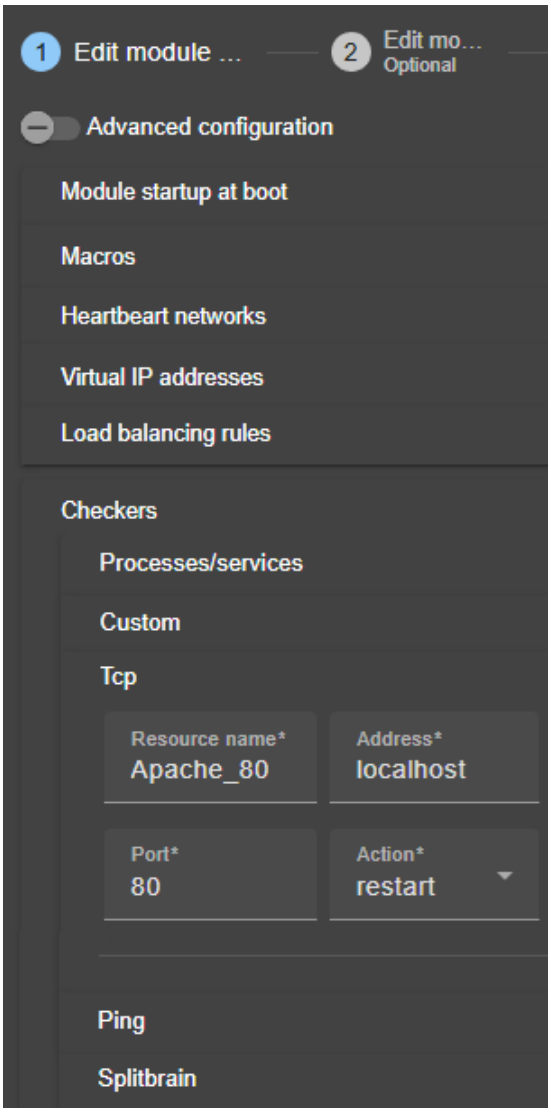
15.5 Exemple de TCP checker

Ci-dessous l'exemple de la configuration d'un TCP checker dans un module ferme. Ce checker teste que la connexion au service web local sur le port 80. Si la connexion échoue, le checker met la ressource **`tcp`**`.Apache_80` à `down`. La règle de failover associée, nommée **`t_`**` Apache_80`, exécute un `restart` du module quand la ressource passe à `down`.

- Le préfixe du nom de la ressource est **`tcp`**.
- Le préfixe du nom de la règle de failover est **`t_`**
- Le suffixe est la valeur de l'attribut `ident`

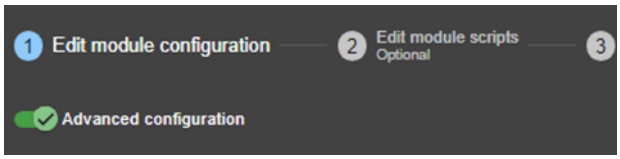
Pour une description des checkers, voir la section 13.10.3.

Pour tester le TCP checker, voir la section 4.4.2 et la section 4.4.3.



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.




```
<!DOCTYPE safe>
<safe>
 <service mode="farm" boot="on">
  <farm>
   <lan name="default"></lan>
  </farm>
  <vip>
   <interface_list>
    <interface check="on">
     <virtual_interface
type="vmac_directed">
      <virtual_addr addr="10.1.0.127"
where="alias" check="on"/>
     </virtual_interface>
    </interface>
   </interface_list>

   <loadbalancing_list>
    <group name="FarmProto">
     <rule port="80" proto="tcp"
filter="on_addr"/>
    </group>
   </loadbalancing_list>
  </vip>

  <user></user>

  <check>
   <tcp ident="Apache_80" when="both"
action="restart">
    <to addr="localhost" port="80"/>
   </tcp>
  </check>
 </service>
</safe>
```

Pour des détails sur la configuration XML de `<tcp>`, voir section 13.11.

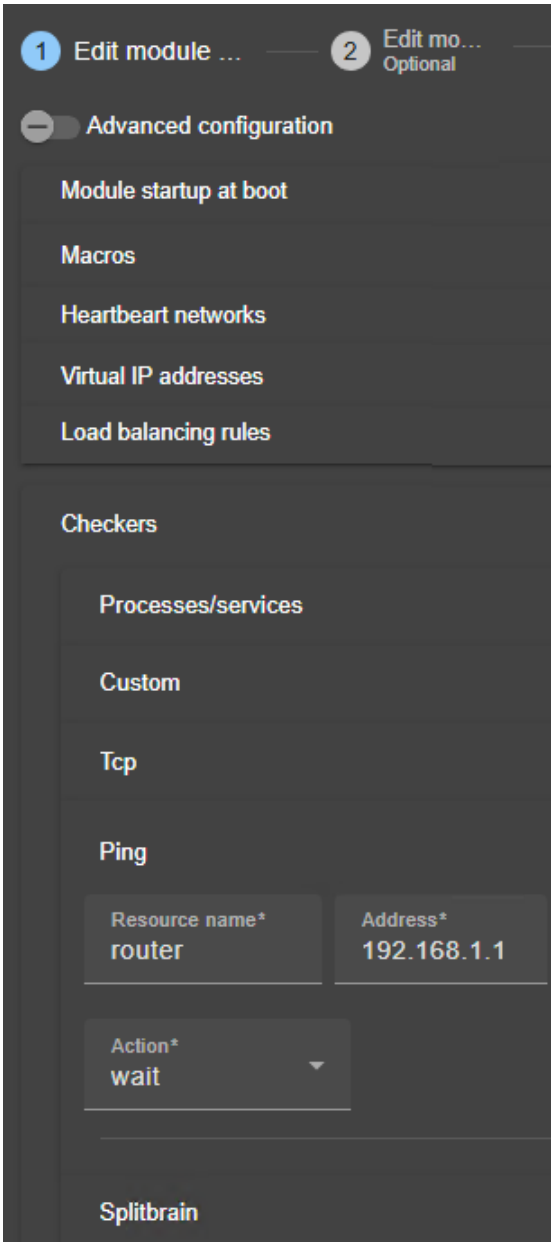
15.6 Exemple de ping checker

Ci-dessous l'exemple de la configuration d'un ping checker dans un module miroir. Ce checker teste si l'adresse `192.168.1.1` répond au ping. Si le ping échoue, le checker affecte la ressource **`ping`**`.router` à `down`. La règle de failover associée, nommée **`p`**`_router`, exécute un `wait` du module quand la ressource passe à `down`.

- Le préfixe du nom de la ressource est **`ping`**.
- Le préfixe du nom de la règle de failover est **`p_`**
- Le suffixe est la valeur de l'attribut `ident`

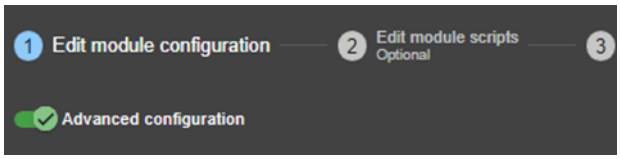
Pour une description des checkers, voir la section 13.10.3.

Pour tester le ping checker, voir la section 4.4.5.



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.




```
<!DOCTYPE safe>
<safe>
<service mode="mirror" boot="on">

 <heart>
  <heartbeat name="default">
  </heartbeat>
 </heart>

 <vip>
  <interface_list>
   <interface check="on">
    <real_interface>
     <virtual_addr addr="10.1.0.126"
where="one_side_alias" check="on"/>
    </real_interface>
   </interface>
  </interface_list>
 </vip>

 <rfs>
  <replicated dir="e:\repdir"/>
 </rfs>

 <user></user>
 <check>
  <ping ident="router" when="pre"
action="wait">
    <to addr="192.168.1.1"/>
  </ping>
 </check>
</service>
</safe>
```

Pour des détails sur la configuration XML de `<ping>`, voir section 13.12.

15.7 Exemple de custom checker avec `customchecker.safe`

Le module `customchecker.safe` est un module miroir de démonstration incluant un cutom checker qui teste la présence d'un fichier sur le serveur primaire**.** Cette fonctionnalité est disponible également dans un module ferme.

Si le fichier est absent, le checker affecte la ressource **`custom`**`. checkfile` à `down`. La règle de failover associée, nommée **`c_`**`checkfile`, exécute un `restart` du module quand la ressource passe à `down`.

- Le préfixe du nom de la ressource est **`custom`**.
- Le préfixe du nom de la règle de failover est **`c_`**
- Le suffixe est la valeur de l'attribut `ident`

Le module `customchecker.safe` est livré avec le package SafeKit et peut servir de base pour l'écriture de votre propre checker.

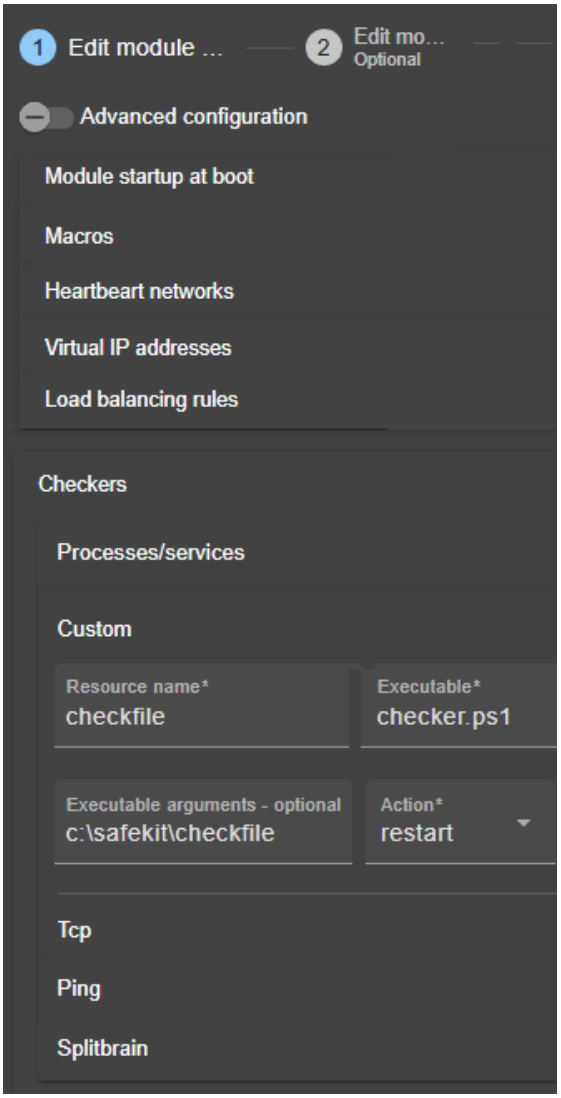
Pour une description des checkers, voir la section 13.10.3.

Pour tester le checker custom, voir la section 4.4.7 et la section 4.4.8.

La description suivante est pour Windows. Pour Linux, veuillez consulter `customchecker.safe` fourni avec le package Linux qui comprend la configuration et les scripts pour Linux.

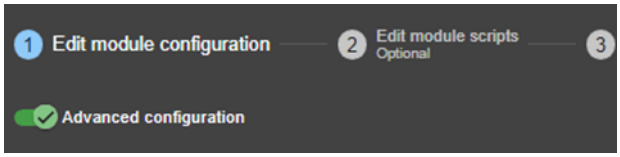
15.7.1 Configuration du module avec un custom checker

L'exemple suivant est la configuration du custom checker prise en charge depuis SafeKit 8 avec l'attribut `action`.



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.




```
<!DOCTYPE safe>
<safe>
<service mode="mirror" boot="on">

 <heart>
  <heartbeat name="default">
  </heartbeat>
 </heart>

 <vip>
  <interface_list>
   <interface check="on">
    <real_interface>
     <virtual_addr addr="10.1.0.126"
where="one_side_alias" check="on"/>
    </real_interface>
   </interface>
  </interface_list>
 </vip>

 <rfs>
  <replicated dir="e:\repdir"/>
 </rfs>

 <user></user>
 <check>
   <custom ident="checkfile"
exec="checker.ps1"
arg="c:\safekit\checkfile" when="prim"
action="restart"/>
 </check>
</service>
</safe>
```

For details on XML configuration of `<custom>`, see section 13.15.

La configuration du custom checker pour SafeKit < 8 est toujours prise en charge. Dans les versions précédentes, la configuration du custom checker nécessitait de définir une règle de failover explicite dans le fichier userconfig.xml comme suit :

```
<check>
 <custom ident="checkfile" exec="checker.ps1" arg="c:\safekit\checkfile"
when="prim"/>
</check>
<failover>
 <![CDATA[
   c_checkfile: if( custom.checkfile == down ) then restart();
 ]]>
</failover>
```

L'attribut `action` n'est pas défini et sa valeur par défaut est `noaction`.

L'assistant de configuration du module ne présente pas la section `failover`. Basculer en « Configuration avancée » pour l'éditer.

15.7.2 Configuration avancée du script du module checker

Le custom checker est une boucle infinie qui effectue un test et affecte la ressource associée l'état `up` ou `down` en fonction du résultat du test.

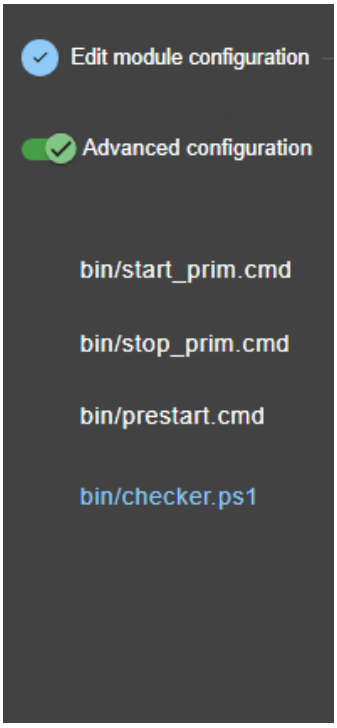
Le checker est automatiquement appelé avec au moins 2 arguments :

- Le 1er argument est le nom module
- Le 2ème est le nom de la ressource à affecter

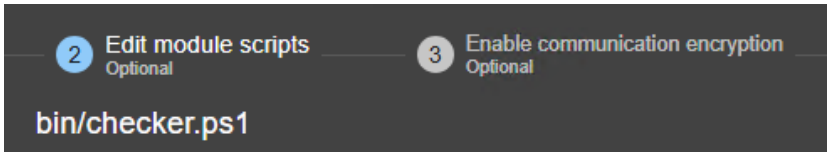
Si la configuration `<custom>` contient l'attribut `arg`, sa valeur est passée comme arguments suivants.

Le checker est écrit en respectant les précautions suivantes :

- La ressource n'est affectée que si sa valeur a changé
- Quand la ressource est `down`, le checker consolide cet état (`grace` fois) avant de l'affecter. Cela peut permettre d'éviter de fausses détections d'erreur.



Basculer en « Configuration avancée » pour lister et éditer tous les scripts.




```
# Custom checker template that tests if a file exists

param([Parameter(Mandatory = $true, ValueFromPipeLine =
$true, position=1)][String]$ModName,
      [Parameter(Mandatory = $true, ValueFromPipeLine =
$true, position=2)][String]$RName,
      [Parameter(Mandatory = $true, ValueFromPipeLine =
$true, position=3)][String]$Arg1Value,
      [Parameter(Mandatory = $false, ValueFromPipeLine
= $false, position=4)][String]$Grace="2",
      [Parameter(Mandatory = $false, ValueFromPipeLine
= $false, position=5)][String] $Period="5"
      )
# return up on success | down on failure
Function test([String]$Arg1Value)
{
  $res="down"
  if (Test-Path "$Arg1Value"){
    $res="up"
  }
  return $res
}

$customchecker=$MyInvocation.MyCommand.Name
$safekit="$env:SAFE/safekit.exe"
$gracecount=0
$prevrstate="unknown"
# wait a little
Start-Sleep $Period
```

```
while ($true){
  Start-Sleep $Period
  $rstate = test($Arg1Value)
  if($rstate -eq "down"){
    $gracecount+=1
  }else{
    $gracecount = 0
    if($prevrstate -ne $rstate){
      & $safekit set -r "$RName" -v $rstate -i
$customchecker -m $ModName
      $prevrstate = $rstate
    }
   }
   if($gracecount -ge $Grace){
     if($prevrstate -ne $rstate){
       & $safekit set -r "$RName" -v $rstate -i
$customchecker -m $ModName
       $prevrstate = $rstate
      }
      $gracecount = 0
   }
}
```

Note the call to `safekit set -r custom.checkfile -v up` (or `down`) to assign the resource value.

Refer to section 9.3 for the description of this command.

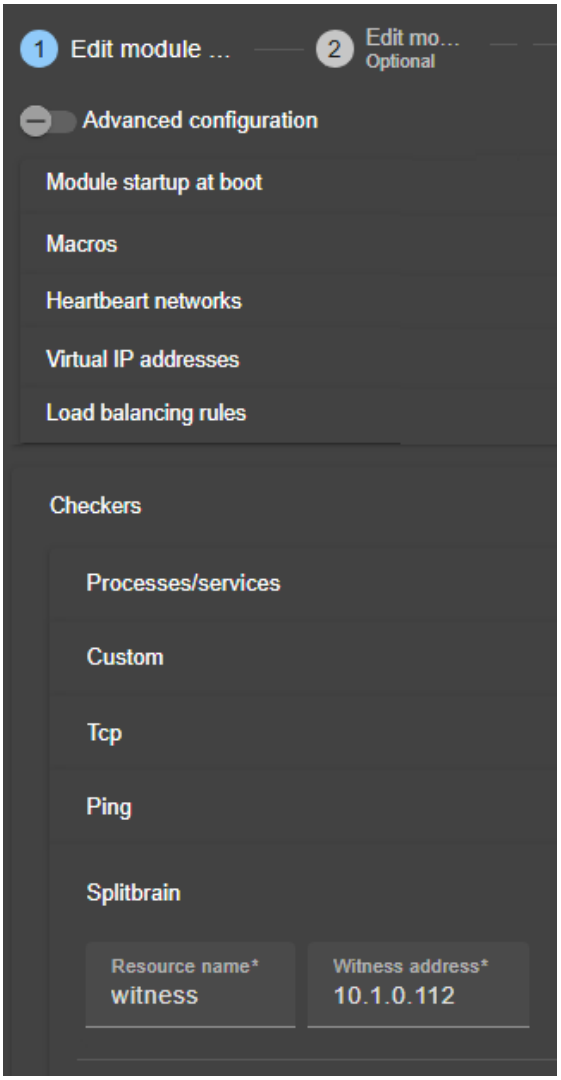
15.8 Exemple de splitbrain checker

Ci-dessous l'exemple de la configuration d'un splitbrain, configurable uniquement dans un module miroir. Ce checker teste si l'adresse `192.168.1.1` répond au ping. Si le ping échoue, le checker affecte la ressource **`splitbrain`**`.witness` à `down`.

En cas d'isolation réseau entre les nœuds, le splitbrain checker affecte la ressource `splitbrain.uptodate` à `up` ou `down` en accord avec l'accès au witness. La règle de failover statique et prédéfinie, nommée **`splitbrain_failure`**, exécute un `wait` du module quand cette ressource passe à `down`. Cela garantit que seul le nœud ayant accès au witness devient `ALONE`, l'autre se bloquant dans l'état `WAIT`.

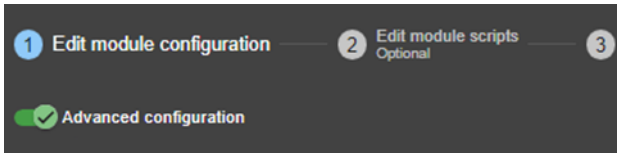
- Le préfixe du nom de la ressource est **`splitbrain`**.
- Le suffixe est la valeur de l'attribut `ident`
- La règle de failover est statique et prédéfinie, et teste l'autre ressource affectée par le splitbrain checker, `splitbrain.uptodate`. Elle se nomme `splitbrain_failure`.

Pour une description des checkers, voir la section 13.10.3.



Pour lancer l'assistant de configuration du module, voir section 3.3.

Basculer en « Configuration avancée » pour éditer le XML si besoin.




```
<!DOCTYPE safe>
<safe>
<service mode="mirror" boot="on">

 <heart>
  <heartbeat name="default">
  </heartbeat>
 </heart>

 <vip>
  <interface_list>
   <interface check="on">
    <real_interface>
     <virtual_addr addr="10.1.0.126"
where="one_side_alias" check="on"/>
    </real_interface>
   </interface>
  </interface_list>
 </vip>

 <rfs>
  <replicated dir="e:\repdir"/>
 </rfs>

 <user></user>

 <check>
  <check>
    <splitbrain ident="witness"
exec="ping" arg="10.1.0.112"/>
  </check>
 </check>
</service>
</safe>
```

Pour des détails sur la configuration XML de `<splitbrain>`, voir section 13.17.

15.9 Exemples de module checker

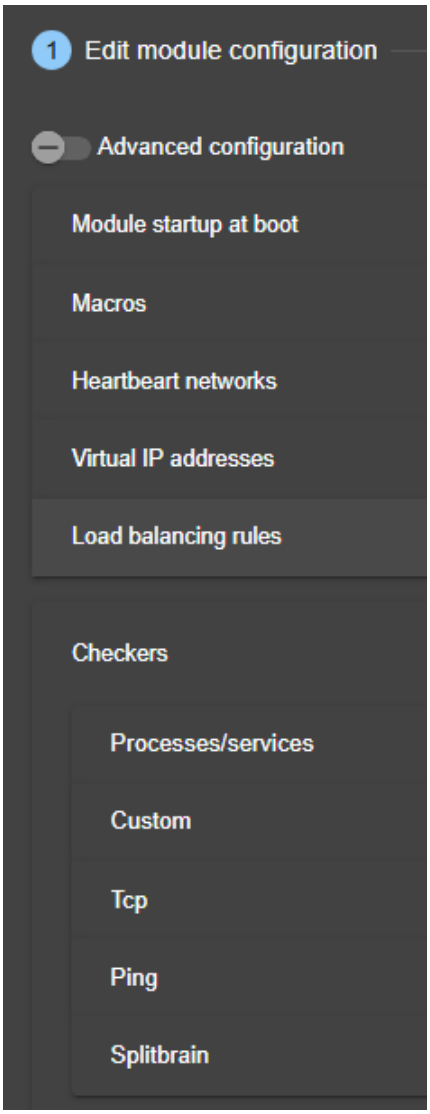
15.9.1 Exemple d'un module ferme dépendant d'un module miroir

Ci-dessous un exemple de la configuration d'un module checker dans un module ferme. Ce checker teste que le module nommé `mirror` avec l'adresse IP virtuelle `10.0.0.129` est prêt (`ALONE` ou `PRIM`). S'il n'est pas prêt, le checker affecte la ressource **`module`**`.mirror_10.0.0.129` à `down`. La règle de failover statique et prédéfinie, nommée `module_failure`, exécute un `wait` sur le module ferme lorsque cette ressource passe à `down`.

- Le préfixe du nom de la ressource est **`module`**.
- Le suffixe est la valeur des attributs `name` et `addr`
- La règle de failover est statique et prédéfinie, et se nomme `module_failure`

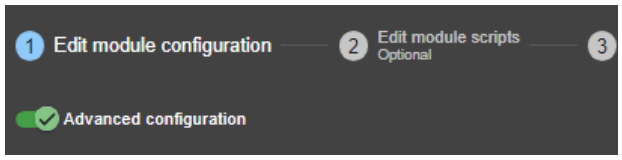
Pour une description des checkers, voir la section 13.10.3.

Pour tester le module checker, voir la section 4.4.6.



Pour lancer l'assistant de configuration du module, voir section 3.3.

L'assistant n'affiche pas le module checker. Basculer en « Configuration avancée » pour le configurer.




```
<!DOCTYPE safe>
<safe>
 <service mode="farm" boot="on">

  <farm>
   <lan name="default"></lan>
  </farm>

  <vip>
   <interface_list>
    <interface check="on">
     <virtual_interface type="vmac_directed">
      <virtual_addr addr="10.1.0.127"
where="alias" check="on"/>
     </virtual_interface>
    </interface>
   </interface_list>

   <loadbalancing_list>
    <group name="FarmProto">
     <rule port="9010" proto="tcp"
filter="on_port"/>
    </group>
   </loadbalancing_list>
  </vip>

  <user></user>

  <check>
   <module name="mirror">
     <to addr="10.0.0.129" port="9010"/>
   </module>
  </check>

 </service>
</safe>
```

Pour des détails sur la configuration XML de `<module>`, voir section 13.16.

La dépendance des modules peut être utilisée lorsque vous déployez des modules ferme et miroir sur le même cluster SafeKit ou lorsque vous déployez des modules ferme et miroir sur deux clusters différents. Dans ce cas, le mot de passe affecté pour initialiser le service web de SafeKit, doit être identique sur les 2 cluster.

15.9.2 Exemple avec `leader.safe` et `follower.safe`

Cet exemple présente 2 modules de démonstration : `leader.safe` et `follower.safe`.

- Le module leader définit les ressources partagées par tous les followers comme l'adresse IP virtuelle ou les répertoires répliqués.
- Les modules followers contiennent le démarrage et l'arrêt de plusieurs applications isolées dans différents modules. Chaque module follower peut être arrêté et démarré indépendamment sans que les autres modules soient arrêtés et sans déconfigurer les ressources du module leader.

Le module leader est un miroir : il inclut dans ses scripts le démarrage/arrêt des modules followers.

Chaque module follower est un module light avec les scripts de démarrage/arrêt d'une application et la détection d'erreur. Chaque module follower est dépendant des défaillances du module leader avec le checker de module suivant :

`follower/conf/userconfig.xml` - voir section 13

```
…
<check>
  <module name="leader"/>
</check>
…
```

Il s'agit d'une configuration synthétique à la place de :

```
…
<check>
  <module name="leader">
    <to addr="127.0.0.1" port="9010"/>
  </module>
</check>
…
```

Si vous avez modifié le port d'écoute du service web de SafeKit (voir section 10.7), remplacez la configuration synthétique par la configuration complète et changez la valeur du port.

15.10 Exemple de checker d'interface réseau

Ci-dessous, l'exemple d'une configuration d'interface checker générée automatiquement lorsque l'option `<interface check="on">` est définie (voir section 13.5).

Dans le fichier de configuration d'un module miroir par exemple, l'adresse virtuelle est définie comme suit :

```
<vip>
  <interface_list>
    <interface check="on">
        <real_interface>
```

```
          <virtual_addr addr="10.0.0.129" where="one_side_alias" check="on"/>
        </real_interface>
    </interface>
  </interface_list>
</vip>
```

Lors de la configuration du module, SafeKit génère la configuration correspondante pour l'interface checker. Pour l'exemple, la configuration automatiquement générée est :

```
<check>
  <intf when="pre" ident="10.0.0.0">
    <to local_addr="10.0.0.107"/>
  </intf>
</check>
```

où la valeur de `ident` correspond au réseau associé à l'adresse IP virtuelle ; la valeur de `local_addr` est la première adresse IP du réseau correspondant à l'adresse virtuelle.

Le checker vérifie que le câble Ethernet est connecté à cette interface. Si le câble est déconnecté, le checker met la ressource associée `intf.10.0.0.0` à `down`. La règle de failover statique et prédéfinie, nommée `interface_failure`, exécute un `wait` sur le module lorsque cette ressource passe à `down`.

- Le préfixe du nom de la ressource est **`intf`**.
- Le suffixe est la valeur de l'attribut `ident`
- La règle de failover est statique et prédéfinie, et se nomme `interface_failure`

Pour plus de détails sur la configuration de checker d'interface, voir la section 13.13.

Pour une description des checkers, voir la section 13.10.3.

Pour tester le checker d'interface réseau, voir la section 4.4.4.

15.11 Exemple d'IP checker

Ci-dessous, l'exemple d'une configuration d'IP checker générée automatiquement lorsque l'option `<virtual_addr check="on" …>` est positionnée (voir section 13.5).

Dans le fichier de configuration d'un module miroir par exemple, l'adresse virtuelle est définie comme suit :

```
<vip>
  <interface_list>
    <interface check="on">
      <real_interface>
        <virtual_addr addr="10.0.0.129" where="one_side_alias" check="on"/>
      </real_interface>
    </interface>
  </interface_list>
</vip>
```

Lors de la configuration du module, SafeKit génère la configuration correspondante pour l'IP checker. Pour l'exemple, la configuration automatiquement générée est :

```
<check>
  <ip ident="10.0.0.129" when="prim">
   <to addr="10.0.0.129"/>
  </ip>
</check>
```

où les valeurs de `ident` et `addr` correspondent à l'adresse IP virtuelle ; la valeur de `when` est `prim` pour un module miroir, `both` pour un module ferme.

Le checker vérifie que l'adresse IP est configurée localement. Si ce n'est pas le cas, le checker affecte la ressource associée `ip.10.0.0.120` à `down`. La règle de failover statique et prédéfinie, nommée `ip_failure`, exécute un `stopstart` sur le module lorsque cette ressource passe à `down`.

- Le préfixe du nom de la ressource est **`ip`**.
- Le suffixe est la valeur de l'attribut `ident`
- La règle de failover est statique et prédéfinie, et se nomme `ip_failure`

Pour plus de détails sur la configuration de checker d'IP, voir la section 13.14.

Pour une description des checkers, voir la section 13.10.3.

15.12 Exemple de notification par mail avec `notification.safe`

Le module `notification.safe` est un module miroir de démonstration pour l'envoi de notifications sur les principaux changements d'état du module. L'exemple suivant propose l'envoi d'un courrier électronique, mais vous pouvez le remplacer par tout autre mécanisme de notification. Sous Windows, il utilise la commande `Send-MailMessage` de l'utilitaire Microsoft Powershell. Sous Linux, il utilise la commande `mail`.

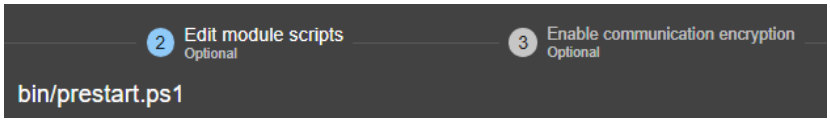
La description suivante est pour Windows. Pour Linux, veuillez consulter `notification.safe` fourni avec le package Linux qui comprend la configuration et les scripts pour Linux.

15.12.1 Notification sur démarrage et arrêt du module

Notification sur démarrage

Le script `prestart` envoie un courriel avec le nom du module et du serveur sur lequel le module est démarré.

Basculer en « Configuration avancée » pour lister et éditer tous les scripts.




```
# Script called on module start for stopping
applications
# before setting SafeKit resources
echo "Running prestart $args"

try{
    $sub    = (Get-Item env:SAFEUSERBIN).Value
    $safebin    = (Get-Item env:SAFEBIN).Value
    $module    = (Get-Item env:SAFEMODULE).Value
    $action = $args[2]
    $retval = 0
    $hostname=(Get-Item env:computername).Value

    if ( $action -eq "start" ) {
         echo "*** Start of module $module on
$hostname"
        # insert here your notification: the module is
starting
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Start of module $module
on $hostname' -Body 'Running prestart'
    }

    #graceful stop
    if (Test-Path -Path "$sub/stop_second.*") { &
"$sub/stop_second" }
    if (Test-Path -Path "$sub/stop_prim.*") { &
"$sub/stop_prim" }
    if (Test-Path -Path "$sub/stop_both.*") { &
"$sub/stop_both" }

    #force stop
    if (Test-Path -Path "$sub/stop_second.*") { &
"$sub/stop_second" force }
    if (Test-Path -Path "$sub/stop_prim.*") { &
"$sub/stop_prim" force }
    if (Test-Path -Path "$sub/stop_both.*") { &
"$sub/stop_both" force }

}catch{
    $retval=-1
}finally{
    echo "prestart exit ($retval)"
    exit $retval
}
```

Note: le “graceful stop” et “force stop” est le contenu standard du script `prestart`.

Notification sur arrêt

Lorsque le module s'arrête, il peut envoyer une notification via le script `poststop`. Celui-ci n'est pas fourni par défaut.

Basculer en « Configuration avancée » pour lister et éditer tous les scripts.

2 Edit module scripts Optional — 3 Enable communication encryption Optional

bin/poststop.ps1

```
# Script called on module stop
# after resetting SafeKit resources
echo "Running poststop $args"

try{
    $module    = (Get-Item env:SAFEMODULE).Value
    $hostname=(Get-Item env:computername).Value
    $action = $args[2]
    $retval = 0
    if ( $action -eq "stop" ) {
         echo "*** Stop of module $module on $hostname"
        # insert here your notification: the module is
stopping
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Stop of module $module
on $hostname' -Body 'Running poststop'
    }

}catch{
    $retval=-1
}finally{
    echo "poststop exit ($retval)"
    exit $retval
}
```

15.12.2 Notification sur changement d'état du module

Le script `transition` peut être utilisée pour envoyer un courriel sur les principaux changements d'état local du module. Par exemple, il peut être utile de savoir quand le module miroir devient `ALONE` (lors d'un basculement par exemple). Le script `transition` n'est pas fourni par défaut.

Basculer en « Configuration avancée » pour lister et éditer tous les scripts.


```
# Script called on module state change
echo "Running transition $args"

try{
    $module    = (Get-Item env:SAFEMODULE).Value
    $hostname=(Get-Item env:computername).Value
    $from = $args[0]
    $to = $args[1]
    $retval = 0

    if ( $from -eq "WAIT" -and $to -eq "ALONE" ) {
         echo "*** Start ALONE of $module on $hostname"
        # insert here your notification: the module is
starting as ALONE
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Start ALONE of module
$module on $hostname' -Body 'Running prestart'
    }
    if ( $from -eq "WAIT" -and $to -eq "PRIM" ) {
         echo "*** Start PRIM of $module on $hostname"
        # insert here your notification: the module is
starting as PRIM
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Start PRIM of module
$module on $hostname' -Body 'Running prestart'
    }
    if ( $from -eq "WAIT" -and $to -eq "SECOND" ) {
         echo "*** Start SECOND of $module on
$hostname"
        # insert here your notification: the module is
starting as SECOND
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Start SECOND of module
$module on $hostname' -Body 'Running prestart'
    }
    if ( $from -ne "WAIT" -and $to -eq "ALONE" ) {
         echo "*** Go ALONE of module $module on
$hostname"
        # insert here your notification: the module is
going ALONE
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Go ALONE of module
$module on $hostname' -Body 'Running prestart'
    }
    if ( $from -ne "WAIT"  -and $to -eq "PRIM" ) {
         echo "*** Go PRIM of module $module on
$hostname"
        # insert here your notification: the module is
going PRIM
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Go PRIM of module
$module on $hostname' -Body 'Running prestart'
    }
```

```
    if ( $from -ne "WAIT"  -and $to -eq "SECOND" ) {
         echo "*** Go SECOND of module $module on
$hostname"
        # insert here your notification: the module is
going SECOND
        # Send-MailMessage -From 'SafeKit' -To
'admin@mydomain.com' -Subject 'Go SECOND of module
$module on $hostname' -Body 'Running prestart'
    }
}catch{
    $retval=-1
}finally{
    echo "transition exit ($retval)"
    exit $retval
}
```

Pour un module ferme, changer la valeur des états.

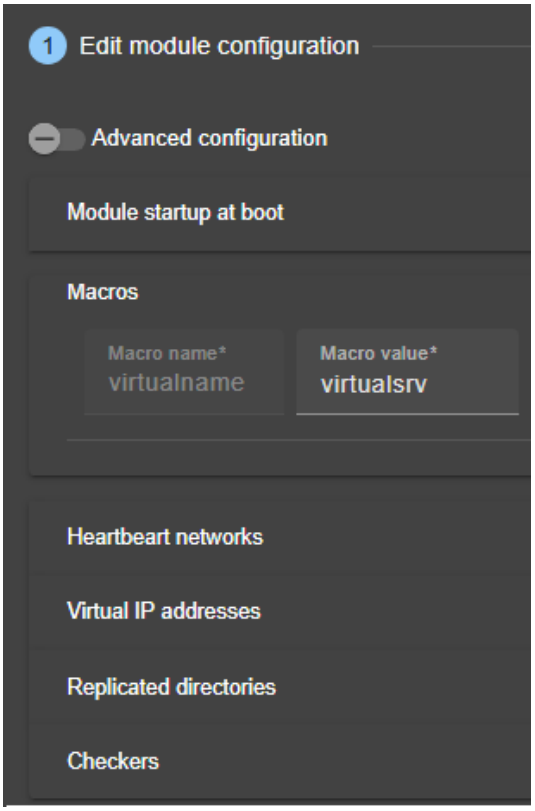
15.13 Exemple d'hostname virtuel avec `vhost.safe`

Le module de démonstration `vhost.safe` montre comment positionner un hostname virtuel dans un module miroir. Cette fonctionnalité est disponible également dans un module ferme. Cette fonctionnalité est disponible également dans un module ferme.

La description suivante est pour Windows. Pour Linux, veuillez consulter `vhost.safe` fourni avec le package Linux qui comprend la configuration et les scripts pour Linux.

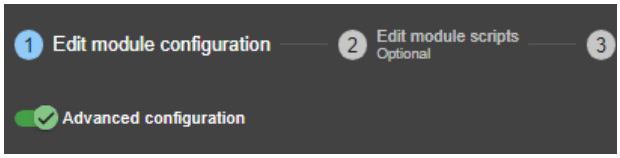
15.13.1 Configuration du module avec un hostname virtuel

Dans l'exemple suivant, une `<macro>` est configurée et sa valeur est utilisée pour définir le nom d'hôte virtuel.



Pour lancer l'assistant de configuration du module, voir section 3.3.

L'assistant n'affiche pas le vhost. Basculer en « Configuration avancée » pour l'éditer.




```
<!DOCTYPE safe>
<safe>
<macro name="virtualname"
value="virtualsrv"/>

<service mode="mirror" boot="on">

 <heart>
  <heartbeat name="default">
  </heartbeat>
 </heart>

 <vip>
  <interface_list>
   <interface check="on">
    <real_interface>
     <virtual_addr addr="10.1.0.126"
where="one_side_alias" check="on"/>
    </real_interface>
   </interface>
  </interface_list>
 </vip>

 <rfs>
  <replicated dir="e:\repdir"/>
 </rfs>

 <user></user>

  <vhost>
   <virtualhostname name="%virtualname%"
envfile="vhostenv.cmd"/>
  </vhost>
</service>
</safe>
```

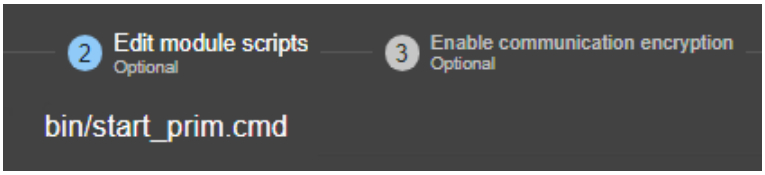
Pour des détails sur la configuration XML de `<module>`, voir section 13.8.

15.13.2 Scripts du module avec un hostname virtuel

En plus de cette configuration, il faut exécuter des commandes spéciales dans les scripts du module.

15.13.2.1 Script start_prim

Le script exécute des commandes pour activer le nom hostname virtuel dans l'environnement du script, ainsi que dans celui du service en Windows.

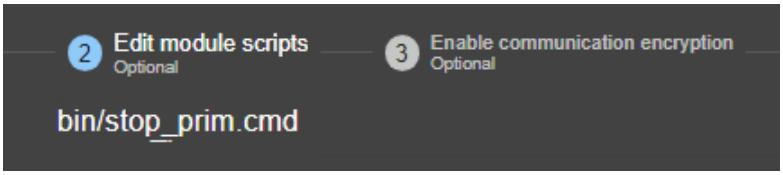



```
@echo off
echo "Running start_prim %*"

rem Set virtual hostname
CALL "%SAFEUSERBIN%\vhostenv.cmd"

rem Next commands use the virtual hostname
FOR /F %%x IN ('hostname') DO SET servername=%%x
echo "hostname is "%servername%

rem WARNING: previous virtual hostname setting is
insufficient to change the hostname for services
rem If one service needs the virtual hostname, you
need also to uncomment the rem following

rem "%SAFE%\private\bin\vhostservice"
SERVICE_TO_BE_DEFINED

set res=0

rem net start "myservice"

set res=%errorlevel%
if %res% == 0 goto end

:stop
"%SAFE%\safekit" printe "start_prim failed"

rem uncomment to stop the module when critical
rem "%SAFE%\safekit" stop -i "start_prim"

:end
```

Pour la description de la commande `safekit`, voir la section 9.

15.13.2.2 Script stop_prim

Le script exécute des commandes pour désactiver le hostname virtuel dans l'environnement du script, ainsi que dans celui du service en Windows.




```
@echo off
echo "Running stop_prim %*"

rem Reset virtual hostname
CALL "%SAFEUSERBIN%\vhostenv.cmd"

rem Next commands use the real hostname
FOR /F %%x IN ('hostname') DO SET servername=%%x
echo "hostname is "%servername%

set res=0

rem default: no action on forcestop
if "%1" == "force" goto end

rem net stop "myservice" /Y
set res=%errorlevel%

rem If necessary, wait for the stop of the services
rem "%SAFEBIN%\sleep" 10

if %res% == 0 goto end

"%SAFE%\safekit" printe "stop_prim failed"

:end
rem WARNING: if the virtual hostname was set for
services in start_prim.cmd,
rem uncomment the following to restore the real
hostname in last stop phase :

rem "%SAFE%\private\bin\vhostservice"
SERVICE_TO_BE_DEFINED
```

Pour la description de la commande `safekit`, voir la section 9.

16.Cluster SafeKit dans le cloud

Vous pouvez installer, configurer et administrer des modules SafeKit qui s'exécutent sur des serveurs virtuels dans les clouds Microsoft Azure, Amazon AWS et Google GCP plutôt que sur des serveurs physiques sur site. Cela nécessite un minimum de paramétrage du cloud et/ou des serveurs, en particulier pour mettre en œuvre une adresse IP virtuelle.

16.1 Cluster SafeKit dans Amazon AWS

Dans ce qui suit, nous supposons que vous êtes familiers avec :

- Amazon Elastic Compute Cloud (Amazon EC2) qui offre des capacités de calcul dans le cloud Amazon Web Services (AWS). Pour plus d'informations sur les fonctionnalités d'Amazon EC2, consultez la section du produit Amazon EC2.
- AWS CloudFormation qui aide à déployer des instances et des applications dans Amazon EC2. Cela permet de gagner beaucoup de temps et d'efforts d'installation : le temps libéré pour la gestion des ressources EC2 peut être exploité pour se consacrer aux applications qui s'exécutent dans AWS.

Avant de pouvoir mettre en œuvre un module SafeKit, l'administrateur doit :

1. Créer des instances (2 pour un module miroir)
2. Effectuer les paramétrages d'AWS, des instances et de SafeKit
3. Enfin, appliquer des configurations SafeKit spécifiques en fonction du module que vous souhaitez mettre en œuvre.

Paramétrage d'AWS

Il faut paramétrer AWS pour :

- associer des adresses publiques à chaque instance si vous souhaitez les administrer avec la console web SafeKit depuis internet
- configurer les groupes de sécurité associés au(x) réseau(x) pour ouvrir les communications du framework SafeKit et de la console web SafeKit. Les ports à ouvrir sont décrits en section 10.3.3.2
- utiliser un réseau à large bande passante et à faible latence si la réplication temps réel est utilisée dans un module miroir

Paramétrage des instances

Sur chaque instance, il faut en plus :

- installer le package SafeKit
- appliquer la configuration HTTPS pour sécuriser la console Web SafeKit (voir section 11)

Paramétrage de SafeKit

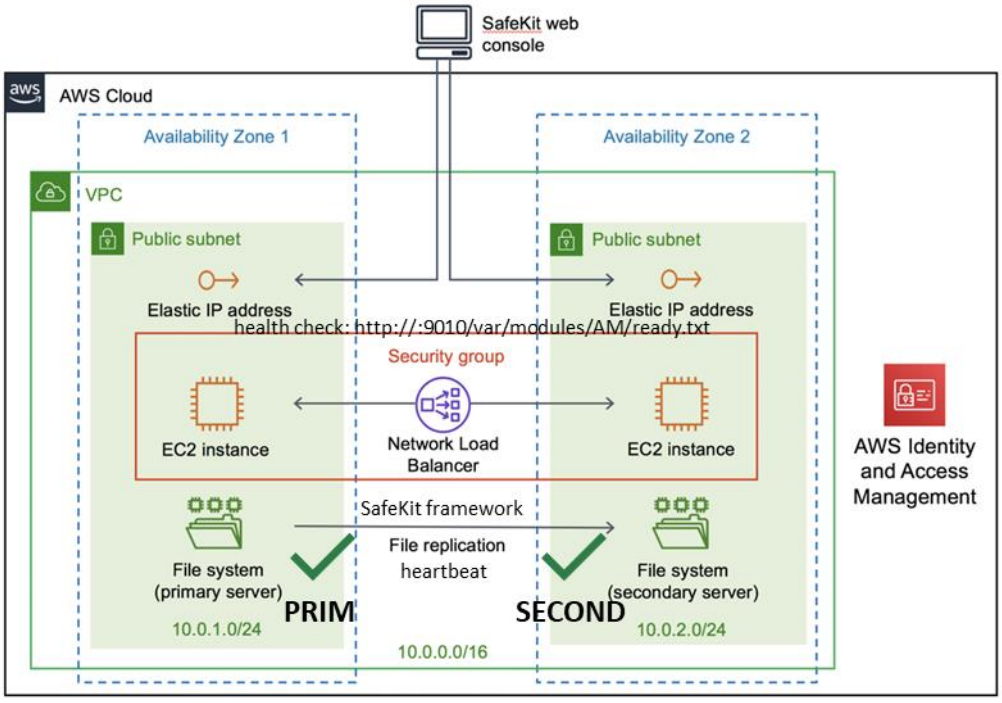
Enfin il faut saisir la configuration du cluster SafeKit et l'appliquer à tous les nœuds (voir section 12). Par exemple, le fichier de configuration du cluster SafeKit serait :

```
<cluster>
<lans>
 <lan name="default">
  <node name="Server1" addr="10.0.11.10"/>
  <node name="Server2" addr="10.0.12.10"/>
 </lan>
</lans>
</cluster>
```

Le lan `default` est utilisé pour les communications du framework SafeKit entre les nœuds du cluster.

16.1.1 Cluster miroir dans AWS

Les fonctionnalités du module miroir sont opérationnelles dans le cloud AWS (réplication de fichiers temps réel, reprise sur panne, détection de mort de processus, checkers, ...), à l'exception du basculement d'adresse IP virtuelle. A la place, vous pouvez configurer un module miroir sur le cluster et utiliser le produit Elastic Load Balancing d'AWS (voir Produits Elastic Load Balancing dans AWS) en le configurant de façon à diriger tout le trafic vers le nœud primaire. L'adresse IP et/ou un nom DNS associés au load balancer, jouent le rôle d'IP virtuelle.



Vous devez configurer vous-même le load balancer et le groupe de sécurité AWS.

Pour le load balancer, vous devez :

- spécifier les règles pour votre application
- définir dans le groupe cible du trafic les nœuds du cluster SafeKit

- définir le test de `vérification de l'état`. Ce test permet de vérifier si l'instance est dans un état sain ou non

Le load balancer achemine le trafic uniquement vers les instances saines. Il reroute le trafic vers l'instance lorsque celle-ci a été restaurée dans un état sain.

SafeKit fournit un testeur de `vérification de l'état` pour chaque module. Vous devez configurer le test de vérification dans le load balancer avec :

- le protocole HTTP
- le port 9010, port du service web de SafeKit
- l'URL `/var/modules/`*`AM`*`/ready.txt` où *`AM`* est le nom du module

Pour un module miroir, le test retourne :

- `OK`, qui signifie l'instance est saine, quand le module est dans l'état ✓`PRIM (Ready)` ou ✓`ALONE (Ready)`
- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états du module

Le groupe de sécurité doit au minimum être configuré pour autoriser les communications pour les protocoles et ports :

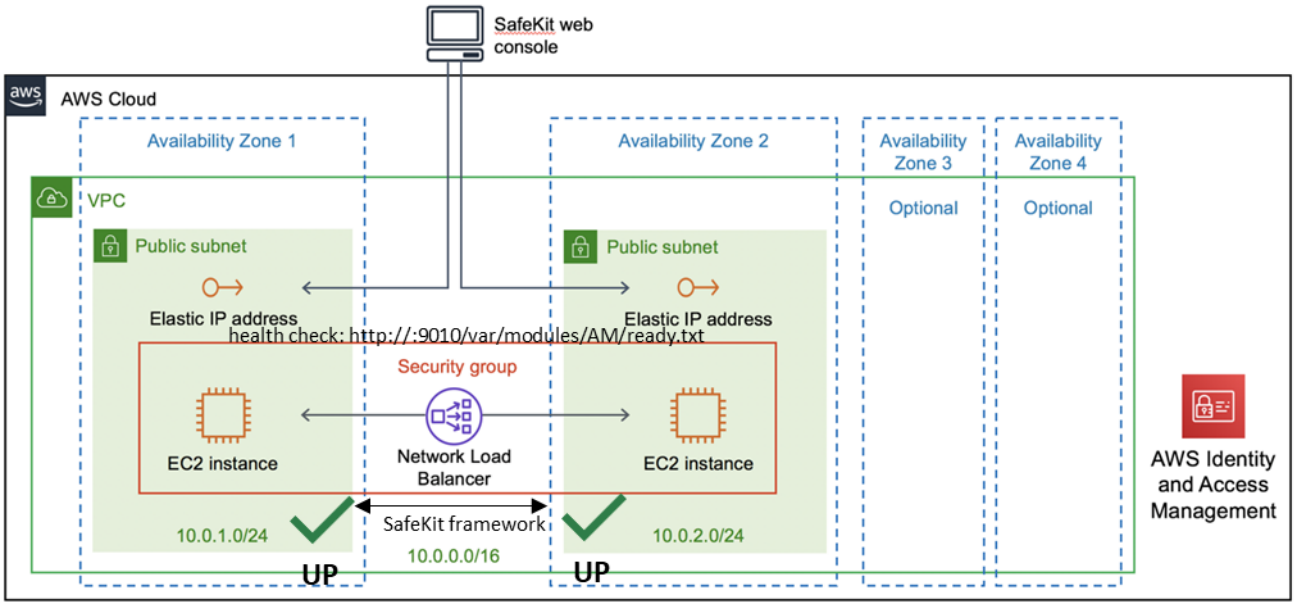
- UDP - 4800 pour le service `safeadmin` (entre les nœuds du cluster SafeKit)
- UDP - 8888 pour le heartbeat du module (entre les nœuds du cluster SafeKit)
- TCP - 5600 pour la réplication temps réelle du module (entre les nœuds du cluster SafeKit)
- TCP - 9010 pour la console web SafeKit en HTTP

  TCP - 9453 pour la console web SafeKit en HTTPS
- TCP - 9001 pour configurer la console web SafeKit en HTTPS

Les valeurs de ports du module dépendent de son id (voir section 10.3.3.2). Les valeurs ci-dessus sont pour le premier module installé.

16.1.2 Cluster ferme dans AWS

La plupart des fonctionnalités du module ferme sont opérationnelles dans le cloud AWS (détection de mort de processus, checker, …), à l'exception du partage de charge sur l'adresse IP virtuelle. A la place, vous pouvez configurer un module ferme sur le cluster et utiliser le produit Elastic Load Balancing d'AWS (voir Produits Elastic Load Balancing dans AWS). L'adresse IP et/ou un nom DNS associés au load balancer, jouent le rôle d'IP virtuelle.



Si vous mettez en place le module ferme en dehors du modèle AWS CloudFormation pour SafeKit, vous devez configurer vous-même le load balancer et le groupe de sécurité AWS.

Pour le load balancer, vous devez :

- spécifier les règles pour votre application
- définir comme cibles du trafic les nœuds du cluster SafeKit
- définir le test de `vérification de l'état`. Ce test permet de vérifier si l'instance est dans un état sain ou non

Le load balancer achemine le trafic uniquement vers les instances saines. Il reroute le trafic vers l'instance lorsque celle-ci a été restaurée dans un état sain.

SafeKit fournit un testeur de `vérification de l'état` pour chaque module. Vous devez configurer le test de vérification dans le load balancer avec :

- le protocole HTTP
- le port 9010, port du service web de SafeKit
- l'URL `/var/modules/`*`AM`*`/ready.txt` où *`AM`* est le nom du module

Pour un module ferme, le test retourne :

- `OK`, qui signifie l'instance est saine, quand le module est dans l'état ✓`UP (Ready)`
- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états

Le groupe de sécurité doit au minimum être configuré pour autoriser les communications pour les protocoles et ports :

- UDP - 4800 pour le service `safeadmin` (entre les nœuds du cluster SafeKit)
- TCP - 9010 pour la console web SafeKit en HTTP

- TCP - 9453 pour la console web SafeKit en HTTPS
- TCP - 9001 pour configurer la console web SafeKit en HTTPS

16.2 Cluster SafeKit dans Microsoft Azure

Dans la suite, nous supposons que vous êtes familiers avec Microsoft Azure, qui est un service de cloud computing créé par Microsoft pour créer, tester, déployer et gérer des applications et des services à travers un réseau mondial de centres de données Microsoft. Pour plus d'informations sur les fonctionnalités et l'utilisation d'Azure, voir le portail Microsoft Azure.

Avant de pouvoir mettre en œuvre un module SafeKit, l'administrateur doit :

1. Créer des machines virtuelles (2 pour un module miroir)
2. Effectuer les paramétrages d'Azure, des machines virtuelles et de SafeKit
3. Enfin, appliquer des configurations SafeKit spécifiques en fonction du module que vous souhaitez mettre en œuvre.

Paramétrage d'Azure

Il faut paramétrer Azure pour :

- associer des adresses IP publiques (éventuellement nom DNS) à chaque machine virtuelle si vous souhaitez les administrer avec la console web SafeKit depuis internet
- configurer, si nécessaire, le groupe de sécurité réseau pour ouvrir les communications du framework SafeKit et de la console web SafeKit. Les ports à ouvrir sont décrits en section 10.3.3.2
- utiliser un réseau à large bande passante et à faible latence si la réplication temps réel est utilisée dans un module miroir

Paramétrage des machines virtuelles

Sur chaque machine virtuelle, il faut en plus :

- installer le package SafeKit
- appliquer la configuration HTTPS pour sécuriser la console Web SafeKit (voir section 11)

Paramétrage de SafeKit

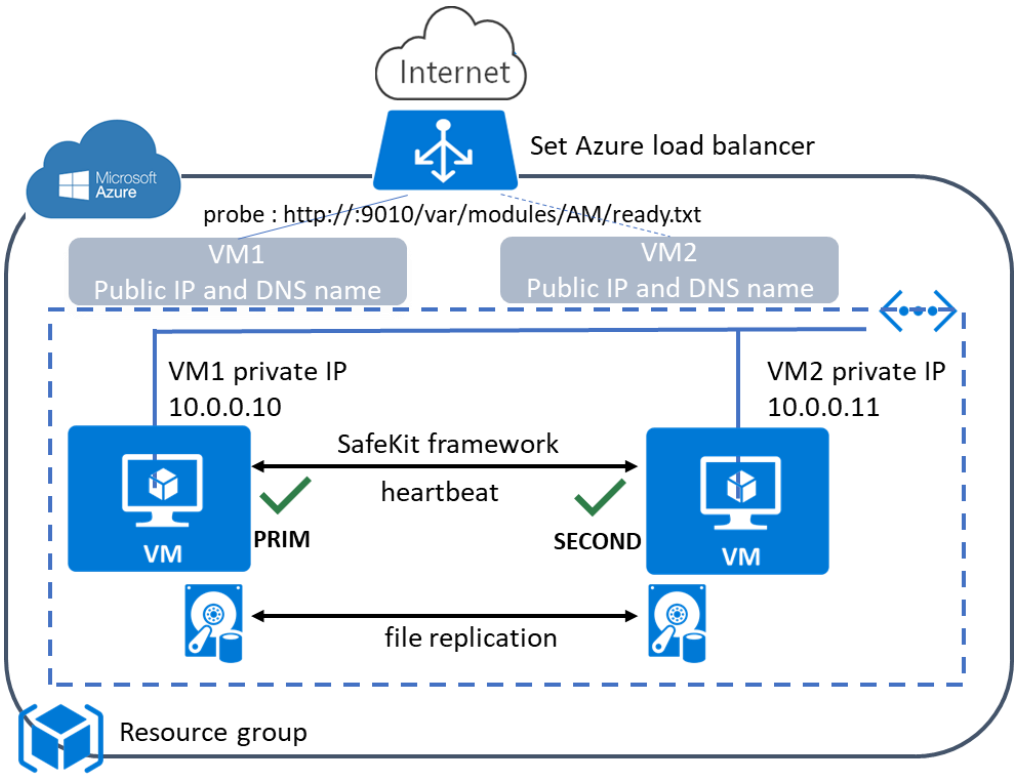
Enfin il faut saisir la configuration du cluster SafeKit et l'appliquer à tous les nœuds (voir section 12). Par exemple, le fichier de configuration du cluster SafeKit serait :

```
<cluster>
<lans>
 <lan name="default">
  <node name="Server1" addr="10.0.0.10"/>
  <node name="Server2" addr="10.0.0.11"/>
 </lan>
</lans>
</cluster>
```

Le lan `default` est utilisé pour les communications du framework SafeKit entre les nœuds du cluster.

16.2.1 Cluster miroir dans Azure

Les fonctionnalités du module miroir sont opérationnelles dans le cloud Azure (réplication de fichiers temps réel, reprise sur panne, détection de mort de processus, checkers, ...), excepté le basculement d'adresse IP virtuelle. A la place, vous pouvez configurer un module miroir sur le cluster et utiliser load balancer proposé par Azure (voir Load Balancer dans Azure) en le configurant de façon à diriger tout le trafic vers le nœud primaire. L'adresse IP associée au load balancer, jouent le rôle d'IP virtuelle.



Vous devez configurer vous-même le load balancer et le groupe de sécurité Azure.

Pour le load balancer, vous devez :

- spécifier les règles pour votre application
- définir dans le backend pool les nœuds du cluster SafeKit
- définir une `sonde d'intégrité`. Cette sonde permet de vérifier si l'instance est dans un état sain ou non

Le load balancer achemine le trafic uniquement vers les instances saines. Il reroute le trafic vers l'instance lorsque celle-ci a été restaurée dans un état sain.

SafeKit fournit une `sonde d'intégrité` pour chaque module. Vous devez configurer la sonde d'intégrité dans le load balancer avec :

- le protocole HTTP
- le port 9010, port du service web de SafeKit
- l'URL `/var/modules/`*`AM`*`/ready.txt` où *`AM`* est le nom du module

Pour un module miroir, le test retourne :

- `OK`, qui signifie l'instance est saine, quand le module est dans l'état ✓`PRIM (Ready)` ou ✓`ALONE (Ready)`
- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états

Le groupe de sécurité réseau doit au minimum être configuré pour autoriser les communications pour les protocoles et ports :

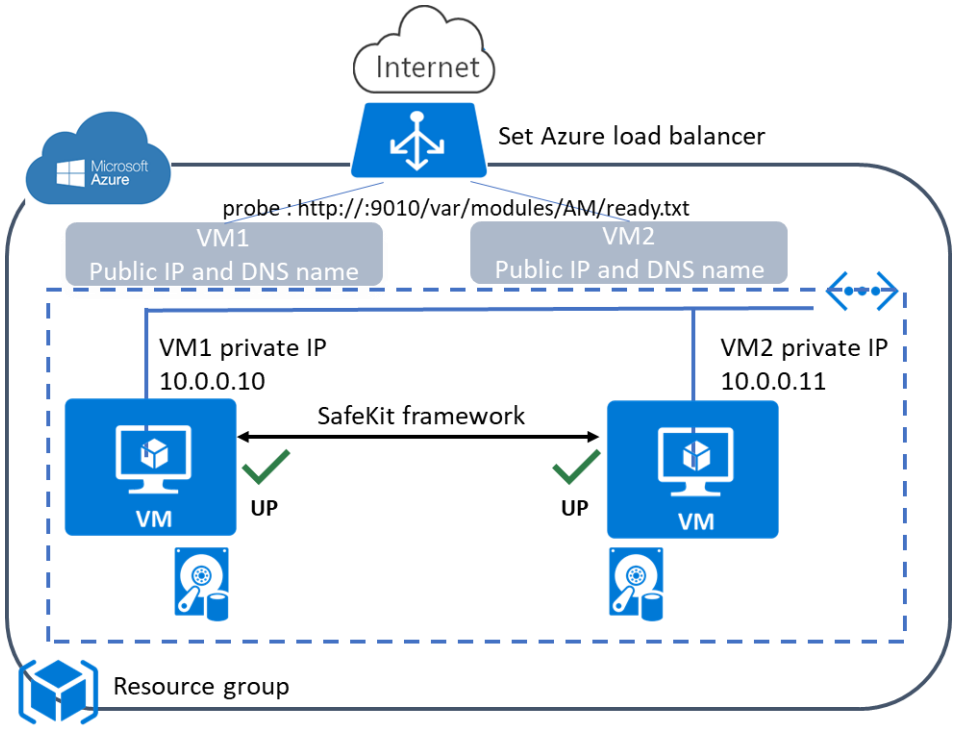
- UDP - 4800 pour le service `safeadmin` (entre les nœuds du cluster SafeKit)
- UDP - 8888 pour le heartbeat du module (entre les nœuds du cluster SafeKit)
- TCP - 5600 pour la réplication temps réelle du module (entre les nœuds du cluster SafeKit)
- TCP - 9010 pour la console web SafeKit en HTTP

  TCP - 9453 pour la console web SafeKit en HTTPS
- TCP - 9001 pour configurer la console web SafeKit en HTTPS

Les valeurs de ports du module dépendent de son id (voir section 10.3.3.2). Les valeurs ci-dessus sont pour le premier module installé.

16.2.2 Cluster ferme dans Azure

La plupart des fonctionnalités du module ferme sont opérationnelles dans le cloud Azure (détection de mort de processus, checker, …), à l'exception du partage de charge sur l'adresse IP virtuelle. A la place, vous pouvez configurer un module ferme sur le cluster et utiliser load balancer proposé par Azure (voir Load Balancer dans Azure). L'adresse IP associée au load balancer, jouent le rôle d'IP virtuelle.



Vous devez configurer vous-même le load balancer et le groupe de sécurité Azure.

Pour le load balancer, vous devez :

- spécifier les règles pour votre application
- définir dans le backend pool les nœuds du cluster SafeKit
- définir une `sonde d'intégrité`. Cette sonde permet de vérifier si l'instance est dans un état sain ou non

Le load balancer achemine le trafic uniquement vers les instances saines. Il reroute le trafic vers l'instance lorsque celle-ci a été restaurée dans un état sain.

SafeKit fournit une `sonde d'intégrité` pour chaque module. Vous devez configurer la `sonde d'intégrité` dans le load balancer avec :

- le protocole HTTP
- le port 9010, port du service web de SafeKit
- l'URL `/var/modules/`*`AM`*`/ready.txt` où *`AM`* est le nom du module

Pour un module ferme, le test retourne :

- `OK`, qui signifie l'instance est saine, quand le module est dans l'état ✓`UP (Ready)`
- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états

Le groupe de sécurité doit au minimum être configuré pour autoriser les communications pour les protocoles et ports :

- UDP - 4800 pour le service `safeadmin` (entre les nœuds du cluster SafeKit)
- TCP - 9010 pour la console web SafeKit en HTTP

  TCP - 9453 pour la console web SafeKit en HTTPS
- TCP - 9001 pour configurer la console web SafeKit en HTTPS

16.3 Cluster SafeKit dans Google GCP

Dans la suite, nous supposons que vous êtes familiers avec Google Cloud Platform (GCP), fournisseur des machines virtuelles (VM) qui s'exécutent dans les centres de données innovants et sur le réseau de fibre optique mondial de Google. Pour plus d'informations sur ses fonctionnalités et son utilisation, voir la documentation Google Cloud Computing.

Avant de pouvoir mettre en œuvre un module SafeKit, l'administrateur doit :

1. Créer des machines virtuelles (2 pour un module miroir)
2. Effectuer les paramétrages de Google Compute Engine (GCP), des machines virtuelles et de SafeKit
3. Enfin, appliquer des configurations SafeKit spécifiques en fonction du module que vous souhaitez mettre en œuvre.

Paramétrage de GCP

Il faut paramétrer le GCP pour :

- associer des adresses publiques (External IP), ou noms DNS, à chaque nœud si vous souhaitez les administrer avec la console web SafeKit depuis internet
- configurer les règles de pare-feu pour le réseau Virtual Private Cloud (VPC) pour ouvrir les communications du framework SafeKit et de la console web SafeKit. Les ports à ouvrir sont décrits en section 10.3.3.2
- utiliser un réseau à large bande passante et à faible latence si la réplication temps réel est utilisée dans un module miroir

Paramétrage des instances

Sur chaque instance, il faut en plus :

- installer le package SafeKit
- appliquer la configuration HTTPS pour sécuriser la console Web SafeKit (voir section 11)

Paramétrage de SafeKit

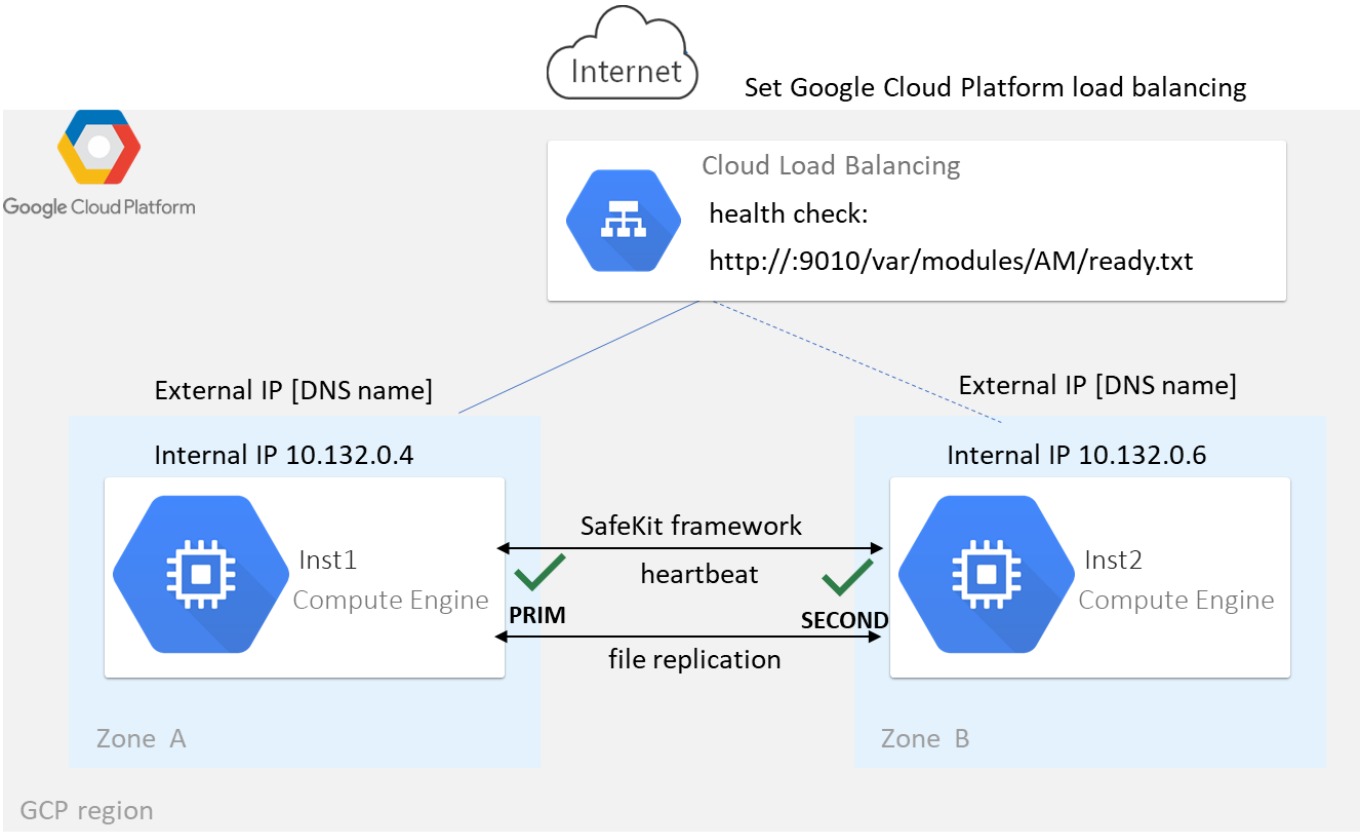
Enfin il faut saisir la configuration du cluster SafeKit et l'appliquer à tous les nœuds (voir section 12). Par exemple, le fichier de configuration du cluster SafeKit serait :

```
<cluster>
<lans>
  <lan name="default">
  <node name="Server1" addr="10.0.11.10"/>
  <node name="Server2" addr="10.0.12.10"/>
 </lan>
</lans>
</cluster>
```

Le lan `default` est utilisé pour les communications du framework SafeKit entre les nœuds du cluster.

16.3.1 Cluster miroir dans GCP

Les fonctionnalités du module miroir sont opérationnelles dans GCP (réplication de fichiers temps réel, reprise sur panne, détection de mort de processus, checkers, ..) à l'exception du basculement d'adresse IP virtuelle. A la place, vous pouvez configurer un module miroir sur le cluster et utiliser le load balancing GCP (voir Load Balancer de GCP) en le configurant de façon à diriger tout le trafic vers le nœud primaire. L'adresse IP associée au load balancer, jouent le rôle d'IP virtuelle.



Si vous mettez en place le module miroir en dehors de la solution du Google Marketplace, vous devez configurer vous-même le load balancer et le pare-feu réseau de Google Cloud Platform.

Pour le load balancer, vous devez :

- spécifier les règles pour votre application
- définir comme cibles du trafic les nœuds du cluster SafeKit
- définir le test de `vérification de l'état`. Ce test permet de vérifier si l'instance est dans un état sain ou non

Le load balancer achemine le trafic uniquement vers les instances saines. Il reroute le trafic vers l'instance lorsque celle-ci a été restaurée dans un état sain.

SafeKit fournit un testeur de `vérification de l'état` pour chaque module. Vous devez configurer le test de vérification dans le load balancer avec :

- le protocole HTTP
- le port 9010, port du service web de SafeKit
- l'URL `/var/modules/`*`AM`*`/ready.txt` où *`AM`* est le nom du module

Pour un module miroir, le test retourne :

- `OK`, qui signifie l'instance est saine, quand le module est dans l'état ✓`PRIM (Ready)` ou ✓`ALONE (Ready)`

- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états

Le pare-feu du réseau doit au minimum être configuré pour autoriser les communications pour les protocoles et ports :

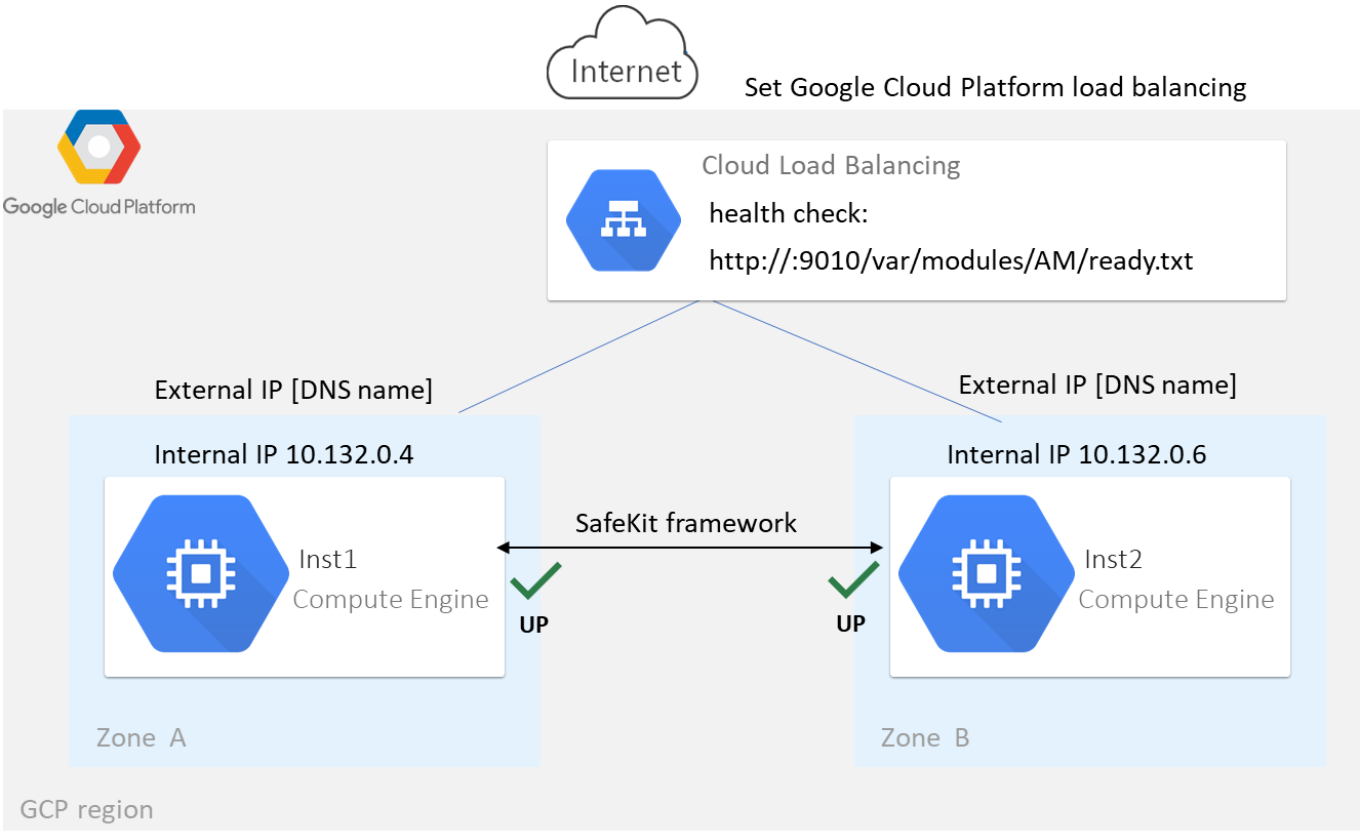
- UDP - 4800 pour le service `safeadmin` (entre les nœuds du cluster SafeKit)
- UDP - 8888 pour le heartbeat du module (entre les nœuds du cluster SafeKit)
- TCP - 5600 pour la réplication temps réelle du module (entre les nœuds du cluster SafeKit)
- TCP - 9010 pour la console web SafeKit en HTTP

  TCP - 9453 pour la console web SafeKit en HTTPS
- TCP - 9001 pour configurer la console web SafeKit en HTTPS

Les valeurs de ports du module dépendent de son id (voir section 10.3.3.2).
Les valeurs ci-dessus sont pour le premier module installé.

16.3.2 Cluster ferme dans GCP

La plupart des fonctionnalités du module ferme sont opérationnelles dans GCP (détection de mort de processus, checker, …), à l'exception du partage de charge sur l'adresse IP virtuelle. A la place, vous pouvez configurer un module ferme sur le cluster et utiliser le load balancing GCP (voir Load Balancer de GCP). L'adresse IP associée au load balancer, jouent le rôle d'IP virtuelle.



Si vous mettez en place le module ferme en dehors de la solution du Google Marketplace, vous devez configurer vous-même le load balancer et le pare-feu réseau de Google Cloud Platform.

Pour le load balancer, vous devez :

- spécifier les règles pour votre application
- définir comme cibles du trafic les nœuds du cluster SafeKit
- définir le test de `vérification de l'état`. Ce test permet de vérifier si l'instance est dans un état sain ou non

Le load balancer achemine le trafic uniquement vers les instances saines. Il reroute le trafic vers l'instance lorsque celle-ci a été restaurée dans un état sain.

SafeKit fournit un testeur de `vérification de l'état` pour chaque module. Vous devez configurer le test de vérification dans le load balancer avec :

- le protocole HTTP
- le port 9010, port du service web de SafeKit
- l'URL `/var/modules/`*`AM`*`/ready.txt` où *`AM`* est le nom du module

Pour un module ferme, le test retourne :

- `OK`, qui signifie l'instance est saine, quand le module est dans l'état ✓`UP (Ready)`
- `NOT FOUND`, qui signifie que l'instance est hors service, dans tous les autres états

Le pare-feu du réseau doit au minimum être configuré pour autoriser les communications pour les protocoles et ports :

- UDP - 4800 pour le service `safeadmin` (entre les nœuds du cluster SafeKit)
- TCP - 9010 pour la console web SafeKit en HTTP
- TCP - 9453 pour la console web SafeKit en HTTPS
- TCP - 9001 pour configurer la console web SafeKit en HTTPS

17.Logiciels tiers

SafeKit apporte les logiciels tiers listés ci-dessous. Pour les détails des licences, se référer aux liens indiqués ou aux fichiers de licence répertoriés sous `SAFE/licenses` (`SAFE=/opt/safekit` en Linux et `SAFE=C:\safekit` en Windows si `%SYSTEMDRIVE%`=C:).

libnet	Packet Construction and Injection Libnet licence - license Utilisé par arpreroute et ping
swagger-ui	https://github.com/swagger-api/swagger-ui Apache2 licence - https://github.com/swagger-api/swagger-ui/blob/master/LICENSE Swagger UI is a collection of HTML, JavaScript, and CSS assets that dynamically generate beautiful documentation from a Swagger-compliant API Utilisé pour visualiser l'API SafeKit
Sqlite3	https://www.sqlite.org/about.html Public Domain licence - https://www.sqlite.org/copyright.html SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine Utilisé par le framework SafeKit

Et uniquement en Windows :

libxml	http://xmlsoft.org MIT licence - http://www.xmlsoft.org/FAQ.html#License Utilisé par le framework SafeKit
libxslt	http://xmlsoft.org/XSLT/ MIT licence - https://gitlab.gnome.org/GNOME/libxslt/blob/master/Copyright Utilisé par le framework SafeKit
Net-SNMP	http://net-snmp.sourceforge.net BSD like and BSD licence - http://www.net-snmp.org/about/license.html Utilisé par l'agent SNMP en Windows
HTTP server	https://httpd.apache.org/ Apache licence - https://www.apache.org/licenses/LICENSE-2.0

	Utilisé par le service web SafeKit pour la console web, les commandes distribuées et le module checker
APR	https://apr.apache.org/ Apache license - https://www.apache.org/licenses/LICENSE-2.0 Utilisé par le serveur HTTP Apache
PCRE	http://www.pcre.org/ BSD license - https://www.pcre.org/licence.txt Utilisé par le serveur HTTP Apache
libexpat	https://github.com/libexpat/libexpat BSD license - https://github.com/libexpat/libexpat/blob/master/expat/COPYING Utilisé par le serveur HTTP Apache
mod_auth_openidc	https://github.com/OpenIDC/mod_auth_openidc Apache2 licence - https://github.com/OpenIDC/mod_auth_openidc/blob/master/LICENSE.txt mod_auth_openidc is an OpenID Certified™ authentication and authorization module for the Apache 2.x HTTP server that implements the OpenID Connect Relying Party Utilisé par le serveur HTTP Apache
cURL	http://curl.haxx.se Curl licence - https://github.com/curl/curl/blob/master/docs/LICENSE-MIXING.md Utilisé par les commandes distribuées et le module checker
OpenSSL	http://www.openssl.org dual OpenSSL and SSLeay licence - https://www.openssl.org/source/license.html Utilisé pour sécurisé la console web, les commandes distribuées et le module checker
Lua	http://www.lua.org MIT licence - https://www.lua.org/license.html Utilisé par le framework et service web SafeKit
Info-ZIP	http://info-zip.org BSD like licence - http://infozip.sourceforge.net/license.html Utilisé pour empaqueté/dépaqueté un .safe

SafeKit utilise les logiciels tiers suivants pour la console Web :

Angular	https://angular.io MIT licence - https://github.com/angular/angular-cli/blob/main/LICENSE Angular is an application-design framework and development platform for creating efficient and sophisticated single-section apps. @angular/animations, @angular/cdk, @angular/common, @angular/core, @angular/forms, @angular/material, @angular/material-moment-adapter, @angular/platform-browser, @angular/router
jszip	https://stuk.github.io/jszip/ MIT OR GPL-3.0-or-later licence - https://github.com/Stuk/jszip/blob/main/LICENSE.markdown A library for creating, reading, and editing .zip files with JavaScript, with a lovely and simple API.
material-icons	https://github.com/marella/material-icons Apache-2.0 licence - https://github.com/marella/material-icons/blob/main/LICENSE
moment	https://github.com/urish/angular-moment#readme MIT licence - https://github.com/urish/angular-moment?tab=MIT-1-ov-file
ngx-logger	https://github.com/dbfannin/ngx-logger#readme MIT licence - https://github.com/dbfannin/ngx-logger?tab=MIT-1-ov-file NGX Logger is a simple logging module for angular
rxjs	https://github.com/ReactiveX/rxjs Apache2 licence - https://github.com/ReactiveX/rxjs/blob/master/LICENSE.txt Reactive Extensions For JavaScript
tslib	https://www.typescriptlang.org/ 0BSD Copyright (c) Microsoft Corporation Runtime library for typescript
vlq	https://github.com/Rich-Harris/vlq/blob/master/README.md MIT licence - https://github.com/Rich-Harris/vlq/blob/master/LICENSE Convert integers to a Base64-encoded VLQ string, and vice versa
zone.js	https://github.com/angular/zone.js MIT licence - https://angular.io/license Implements Zones for JavaScript

Cette liste est aussi disponible à l’emplacement suivant :
safekit/web/htdocs/console/fr/3rdpartylicenses.txt .

Index des messages du journal du module

"If you are sure that this server has valid data, run safekit prim to force start as primary", 102, 104, 121

"If you are sure that this server has valid data, run safekit primforce to force start as primary", 109

"Reintegration ended (synchronize)", 79

"Updating directory tree from /replicated", 79

Load-balancing

"farm load: 128/256 (group FarmProto_0)" , 113, 85, 86

"farm membership: node1 (group FarmProto_0)", 85, 86

"farm membership: node1 node2 (group FarmProto_0)" , 113, 85, 86

"farm membership: node2 (group FarmProto_0)", 86

"Local state ..."

"Local state ALONE Ready", 101, 77, 82

"Local state PRIM Ready", 101,77

"Local state SECOND Ready",101, 77

"Local state UP Ready",112 ,113

"Local state WAIT NotReady", 121, 106

"Remote state ..."

"Remote state ALONE Ready", 101,82

"Remote state PRIM Ready", 101, 77

"Remote state SECOND Ready",101, 77

"Remote state UNKNOWN Unknown", 80, 82

"Resource ..."

"Resource custom.id set to down by customscript", 95, 96, 121, 122

"Resource custom.id set to up by customscript", 95

"Resource heartbeat.0 set to down by heart", 80, 82

"Resource heartbeat.flow set to down by heart", 80, 82

"Resource intf.ip.0 set to down by intfcheck", 92, 121

"Resource intf.ip.0 set to up by intfcheck", 92

"Resource module.othermodule_ip set to down by modulecheck", 94, 121

"Resource module.othermodule_ip set to up by modulecheck", 94

"Resource ping.id set to down by pingcheck", 93, 121

"Resource ping.id set to up by pingcheck", 93

"Script ..."

"Transition ..."

Autres messages

Index