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***	« Architectures de haute disponibilité » page 15 « Cluster SafeKit dans le cloud page 297
	Installation	« Installation » page 25
	Console	« La console web de SafeKit » page 37 « Sécurisation du service web de SafeKit » page 177
	Configuration avancée	« Cluster.xml pour la configuration du cluster SafeKit » page 205 « Userconfig.xml pour la configuration du module » page 211 « Scripts du module pour la configuration du module » page 269 « Exemples de userconfig.xml et scripts du module » page 275
	Administration	« Administration d'un module miroir » page 95 « Administration d'un module ferme » page 107 « Interface ligne de commande » page 143 « Administration avancée » page 157
	Support	« Tests » page 69 « Résolution de problèmes » page 111 « Accès au support Evidian » page 133 « Index des messages du log page 313
	Autres	« Table des matières » page 5 « Logiciels tiers » page 309
Version	SafeKit 8.2	
OS supportés	Windows et Linux ; pour une liste détaillée des OS supportés, voir ici	

Site web	Site marketing Evidian : http://www.evidian.com/safekit Site support Evidian : https://support.evidian.com/safekit
Ref	39 F2 38MC 02

Si vous avez des commentaires ou des questions relatives à ce document, envoyez-nous s'il vous plaît un courriel à **institute@evidian.com**

Table des matières

1. Architectures de haute disponibilité

1.1 Définition du cluster SafeKit

Un cluster SafeKit est un groupe de serveurs sur lesquels Safekit est installé et en fonctionnement.

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.

La définition du cluster est un prérequis à toute installation et configuration de modules SafeKit. La définition du cluster se fait via la console web comme décrit en section 3.2 page 39. La console web de SafeKit offre la possibilité d'administrer un ou plusieurs clusters SafeKit.

1.2 Définition d'un module SafeKit -intégration d'application

Un module est une personnalisation de SafeKit pour une application. Le module définit la solution de haute disponibilité prévue pour l'application et les procédures de reprise de l'application. Différents modules peuvent être définis pour différentes applications.

Concrètement, un module applicatif inclut :

- un fichier de configuration principal `userconfig.xml` qui définit les réseaux utilisés par les serveurs, les fichiers à répliquer en temps réel (pour un module miroir), la configuration d'adresse IP virtuelle, les critères de partage de charge (pour un module ferme) et plus…
- les scripts de démarrage et d'arrêt de l'application

SafeKit propose deux types de module détaillés dans ce chapitre :

- le module miroir
- le module ferme

Plusieurs modules applicatifs peuvent s'exécuter sur le même cluster de serveurs permettant d'imaginer des architectures avancées :

- active/active : 2 modules miroirs en backup l'un de l'autre
- N-1 : N modules miroirs avec un seul backup
- mixte ferme et miroir : mixte le partage de charge, la réplication de fichiers et la reprise

1.3 Module miroir : réplication temps réel synchrone et reprise sur panne

1.3.1 Réplication de fichiers et reprise sur panne

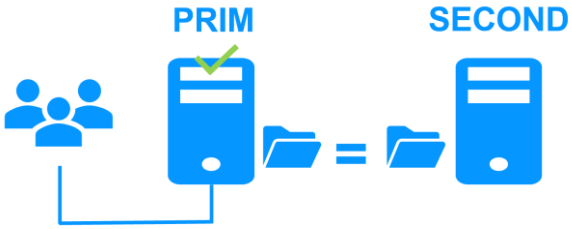
L'architecture miroir est une solution de haute disponibilité de type primaire - secours applicable à n'importe quelle application. L'application est exécutée sur un serveur primaire et redémarrée automatiquement sur un serveur de secours si le serveur primaire est défaillant.

L'architecture miroir peut être configurée avec ou sans réplication de fichiers. Avec la réplication de fichiers, cette architecture 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 et la reprise sur panne se fait sur le serveur secondaire depuis la version la plus à jour des données. Si la disponibilité de l'application est plus critique que la synchronisation des données, la politique par défaut peut être relâchée pour autoriser une reprise sur panne par le serveur secondaire lorsque la date de la dernière synchronisation est inférieure à un délai configurable.

Microsoft SQL Server.safe, MySQL.safe, Oracle.safe sont des exemples de modules applicatifs de type "miroir". Vous pouvez écrire votre propre module miroir pour votre application à partir du module générique Mirror.safe.

Le système de reprise fonctionne de la façon suivante.

1.3.2 Etape 1. Etat normal d'un miroir



Seuls les noms des répertoires de fichiers à répliquer sont configurés dans SafeKit. Il n'y a pas de prérequis sur l'organisation disque des deux serveurs. Les répertoires à répliquer peuvent être localisés dans le disque système.

Le serveur 1 (`PRIM`) exécute l'application.

SafeKit réplique les fichiers ouverts par l'application. Seules les modifications faites par l'application à l'intérieur des fichiers sont répliquées en temps réel à travers le réseau, limitant ainsi le trafic.

Grace à la réplication synchrone des écritures sur les disques des deux serveurs, aucune donnée n'est perdue en cas de panne.

1.3.3 Etape 2. Reprise sur panne

Lorsque le serveur 1 est défaillant, la reprise sur le serveur 2 est assurée. SafeKit bascule l'adresse IP virtuelle du cluster et redémarre automatiquement l'application sur le serveur 2. L'application retrouve les fichiers répliqués par SafeKit avec l'assurance qu'aucune écriture synchrone sur disque n'a été perdue entre le serveur 1 et le serveur 2. L'application continue son exécution sur le serveur 2 en modifiant localement ses fichiers qui ne sont plus répliqués vers le serveur 1.

Le temps de basculement est égal au temps de détection de la panne (time-out configuré à 30 secondes par défaut) et au temps de relance de l'application. Sur la machine secondaire, il n'y a pas de temps lié au remontage du système de fichiers ou au passage des procédures de recovery du système de fichiers, comme avec les solutions de réplication de disques.

1.3.4 Etape 3. Réintégration après panne

A la reprise après panne du serveur 1 (réintégration du serveur 1), SafeKit resynchronise automatiquement les fichiers de ce serveur à partir de l'autre serveur. Seuls les fichiers modifiés sur le serveur 2 pendant l'inactivité du serveur 1 sont resynchronisés.

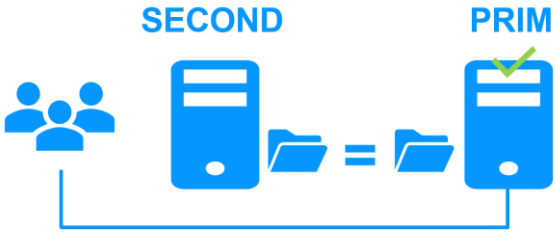
La réintégration du serveur 1 se fait sans arrêter l'exécution des applications sur le serveur 2. Cette propriété est un différentiateur du produit SafeKit par rapport à d'autres solutions qui nécessitent d'arrêter les applications sur le serveur 2 pour réintégrer le serveur 1.

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 la secondaire a crashé (power off) ou a été incorrectement arrêtée (exception du processus de réplication `nfsbox`), 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

5. Si les fichiers sont modifiés sur le serveur primaire ou secondaire alors que SafeKit est arrêté, les répertoires répliqués sont totalement réintégrés sur la secondaire.

1.3.5 Etape 4. Retour à la normale



Après la réintégration, les fichiers sont à nouveau en mode miroir comme à l'étape 1. Le système est en haute disponibilité avec l'application qui s'exécute sur le serveur 2 et avec comme secours le serveur 1. Les modifications de l'application dans les fichiers sont répliquées en temps réel du serveur 2 vers le serveur 1.

Si l'administrateur souhaite que son application s'exécute en priorité sur le serveur 1, il peut exécuter une commande de basculement, soit manuellement à un moment opportun, soit automatiquement par configuration.

1.3.6 Solution de réplication synchrone qui ne perd pas de données en cas de panne

Il existe une grande différence entre réplication synchrone de données mise en œuvre par la solution miroir de SafeKit et réplication asynchrone de données telle qu'elle est traditionnellement mise en œuvre dans les solutions de réplication de fichiers.

Avec une réplication synchrone, lorsqu'une IO disque est réalisée par l'application ou le cache système sur le serveur primaire et sur un fichier répliqué, SafeKit attend l'acquittement de l'IO du disque local et du serveur secondaire avant d'envoyer l'acquittement à l'application ou au cache système.

La réplication synchrone et temps réel des données sur des fichiers ouverts par une application assure la haute disponibilité applicative sans perte de données en cas de panne. Notamment, la réplication synchrone des fichiers assure que toute donnée commitée sur un disque par une application transactionnelle est retrouvée sur la machine secondaire.

La bande passante d'un LAN entre les deux serveurs est nécessaire pour mettre en œuvre une réplication synchrone de données avec éventuellement un LAN étendu dans deux salles machines éloignées de plusieurs kilomètres.

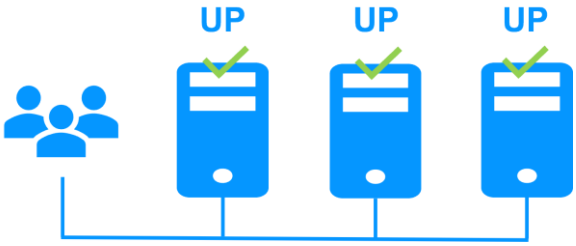
Avec la réplication asynchrone mise en œuvre par d'autres solutions, les IOs sont mises dans une file sur le serveur primaire et les acquittements du serveur secondaire ne sont pas attendus. Donc, toutes les données qui n'ont pas eu le temps d'être recopiées à travers le réseau sur le second serveur sont perdues en cas de panne du premier serveur. Notamment, une application transactionnelle perd des données commitées en cas de panne. La réplication asynchrone est adaptée à la réplication de données à travers un réseau bas débit de type WAN, pour réaliser un backup à distance sur plus de 100 kilomètres.

SafeKit propose une solution asynchrone sans perte de données en assurant l'asynchronisme non pas sur la machine primaire mais sur la machine secondaire. Dans cette solution, SafeKit attend toujours l'acquittement des deux machines avant d'envoyer l'acquittement à l'application ou au cache système. Mais sur la secondaire, il y a 2 options asynchrone ou synchrone. Dans le cas asynchrone (option <rfs async="second">), la secondaire envoie l'acquittement à la primaire dès réception de

l'IO puis écrit sur disque. Dans le cas synchrone (<rfs async="none">), la secondaire écrit l'IO sur disque puis envoie l'acquittement à la primaire. Le mode async="none" est nécessaire si l'on considère une double panne électrique simultanée des deux serveurs avec impossibilité de redémarrer l'ex serveur primaire et obligation de redémarrer sur le secondaire.

1.4 Module ferme : partage de charge réseau et reprise sur panne

1.4.1 Partage de charge réseau et reprise sur panne



L'architecture ferme permet d'assurer à fois le partage de charge réseau, à travers une distribution transparente du trafic réseau et une reprise sur panne matérielle et logicielle. Cette architecture fournit une solution simple au problème de la montée en charge. La même application s'exécute sur chacun des serveurs et la charge est distribuée par répartition de l'activité réseau sur les différents serveurs de la ferme.

L'architecture ferme est adaptée aux applications frontales telles que les services web. Apache_farm.safe, Microsoft IIS_farm.safe sont des exemples de modules applicatifs de type ferme. Vous pouvez écrire votre propre module 'ferme' pour votre application à partir du module générique Farm.safe.

1.4.2 Principe d'une adresse IP virtuelle avec partage de charge réseau

L'adresse IP virtuelle est configurée localement sur chaque serveur de la ferme. Le trafic du réseau à destination de l'adresse IP virtuelle est distribué entre les serveurs grâce à un filtre chargé dans le système d'exploitation de chaque serveur.

L'algorithme de partage de charge dans le filtre est basé sur l'identité des paquets client (adresse IP client, port TCP client). Suivant l'identité du paquet client en entrée, seul un filtre dans un serveur accepte le paquet ; les autres filtres dans les autres serveurs le rejettent. Une fois un paquet accepté par le filtre sur un serveur, seul le CPU et la mémoire de ce serveur sont utilisés par l'application qui répond à la requête du client. Les messages de retour de l'application sont envoyés directement du serveur vers le client.

Lorsqu'un serveur est défaillant, le protocole de gestion du groupe des serveurs en vie reconfigure les filtres pour redistribuer le trafic vers les serveurs disponibles.

1.4.3 Critères de partage de charge pour les services web à état et sans état

Avec un service à état, il y a affinité de session. Le même client doit être connecté sur le même serveur sur plusieurs sessions HTTP/TCP pour retrouver son contexte sur le serveur. Dans ce cas, la règle de load balancing SafeKit est configurée sur l'adresse IP des clients. Ainsi, le même client est toujours connecté sur le même serveur sur plusieurs sessions TCP. Et différents clients sont répartis sur les différents serveurs de la ferme. Cette configuration est à choisir pour les services web à état lorsqu'il y a affinité de sessions.

Avec un service web sans état, il n'y a pas d'affinité de session. Le même client peut être connecté sur des serveurs différents dans la ferme lors de sessions HTTP/TCP successives. Dans ce cas, la règle de load balancing SafeKit est configurée sur l'identité de la session TCP du client. Cette configuration est celle qui répartit le mieux les sessions entre les serveurs mais elle requiert un service TCP sans affinité de session.

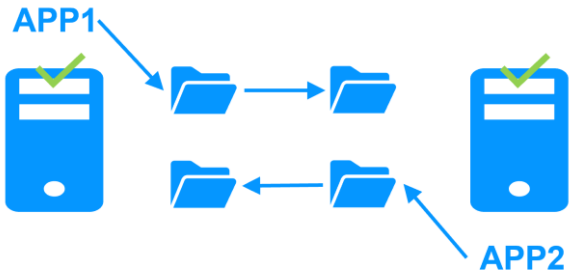
D'autres algorithmes de partage de charge sont proposés pour des services UDP.

1.5 Combiner les modules miroir et ferme

1.5.1 Actif/Actif : 2 modules miroirs en backup l'un de l'autre

Deux serveurs actifs en miroir l'un de l'autre

Dans une architecture active / active, il y a deux serveurs et deux modules applicatifs miroirs en reprise mutuelle (Appli1.Safe et Appli2.Safe). Chaque serveur applicatif est secours de l'autre serveur applicatif.



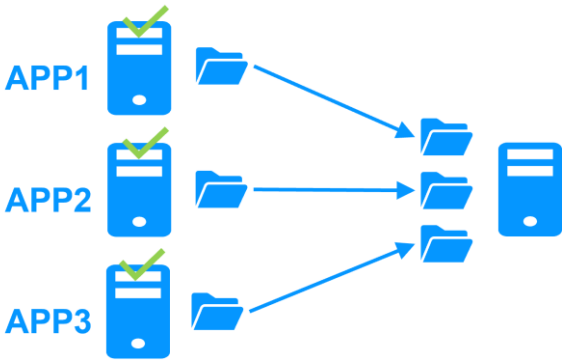
Lorsqu'un serveur applicatif est défaillant, les deux applications sont actives sur le serveur applicatif restant. Et après le redémarrage du serveur défaillant, chaque application est de nouveau active sur son serveur primaire par défaut.

Un cluster en reprise mutuelle est une solution plus économique que deux clusters miroirs. Il n’y a pas de serveur de reprise inactif passant son temps à attendre la panne du serveur primaire et à assurer seulement la reprise applicative. Notez que dans une telle architecture, en cas de défaillance d'un serveur, le serveur restant doit supporter la charge des deux applications.

1.5.2 N-1 : N modules miroirs avec un seul backup

Backup partagé entre plusieurs serveurs actifs

Dans l'architecture N-1, il y a N modules applicatifs de type miroir mis en œuvre sur N serveurs primaires et un seul serveur backup.



Si un des N serveurs applicatifs actifs est défaillant, le serveur de secours redémarre l'application qui tournait sur le serveur défaillant. Quand le serveur défaillant redémarre, l'application bascule du serveur de secours vers son serveur d'origine.

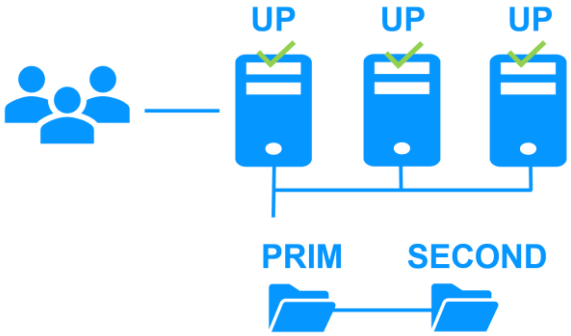
Dans le cas d'une panne, contrairement à l'architecture actif/actif, le serveur de secours n'est pas surchargé par l'exécution de plusieurs applications. Dans le cas particulier de plusieurs pannes simultanées, toutes les applications défaillantes sont redémarrées sur le serveur de secours.

1.5.3 Mixte ferme/miroir : partage de charge réseau, réplication de fichiers et reprise sur panne

Partage de charge réseau, réplication de fichiers et reprise sur panne

Des modules applicatifs ferme et miroir peuvent être mixés sur des serveurs physiques communs.

Cette possibilité permet de mettre en œuvre une architecture applicative multi tiers telle que Apache_farm.safe (ferme avec partage de charge et reprise) et MySQL.safe (miroir avec réplication de fichiers et reprise) sur des serveurs applicatifs communs.



Ainsi, le partage de charge, la réplication de fichiers et la reprise sont mis en œuvre de manière cohérente sur les mêmes serveurs physiques. Ce type d'architecture est propre à SafeKit et unique sur le marché !

1.6 La plus simple solution pour la haute disponibilité dans le cloud

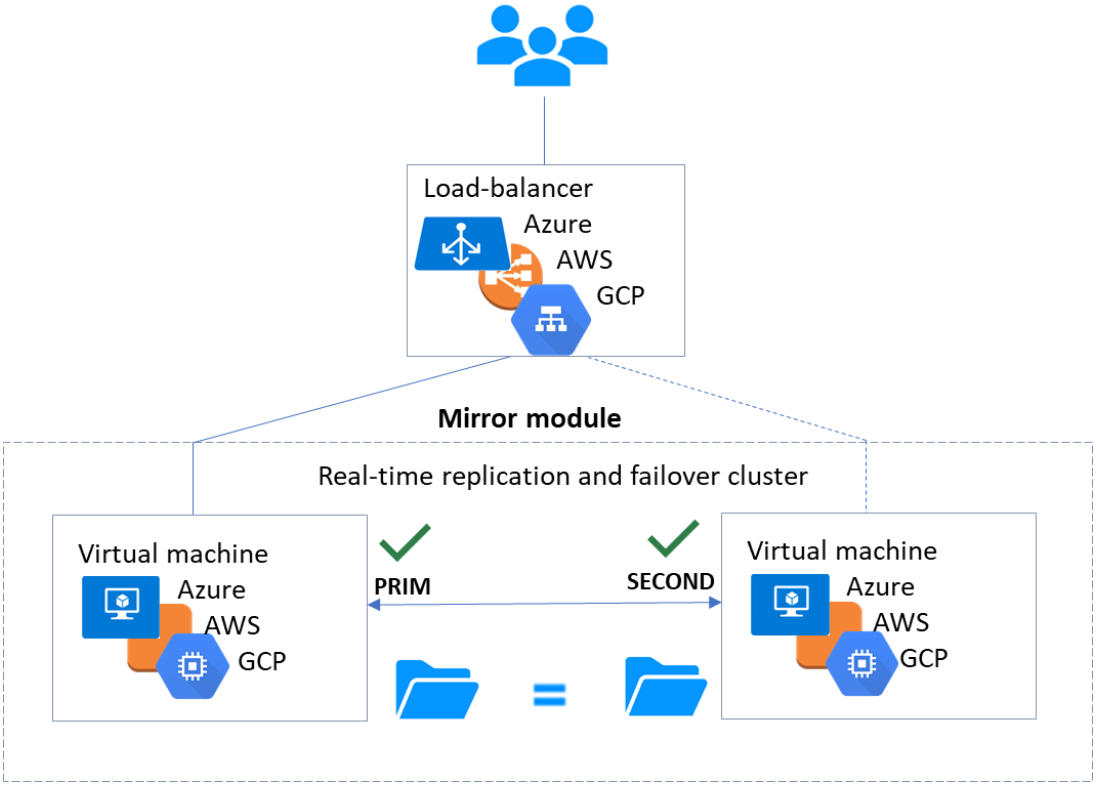
SafeKit est la solution la plus simple pour mettre en œuvre un cluster hautement disponible dans les clouds Microsoft Azure, Amazon AWS et Google GCP. SafeKit peut être implémenté sur des machines virtuelles existantes ou sur une nouvelle infrastructure, que vous créez en cliquant simplement sur un bouton qui déploie et configure tout pour vous dans les clouds Azure ou AWS.

Pour une description complète, voir section 16 page 297.

1.6.1 Cluster miroir dans Microsoft Azure, Amazon AWS et Google GCP

SafeKit est la plus simple solution dans les clouds Azure, AWS et GCP, pour mettre en œuvre un cluster actif-passif avec failover applicatif et réplication temps réel et continue des données (module miroir).

Pour un mise en œuvre rapide, se référer à cluster miroir dans Azure, cluster miroir dans AWS ou cluster miroir dans GCP.



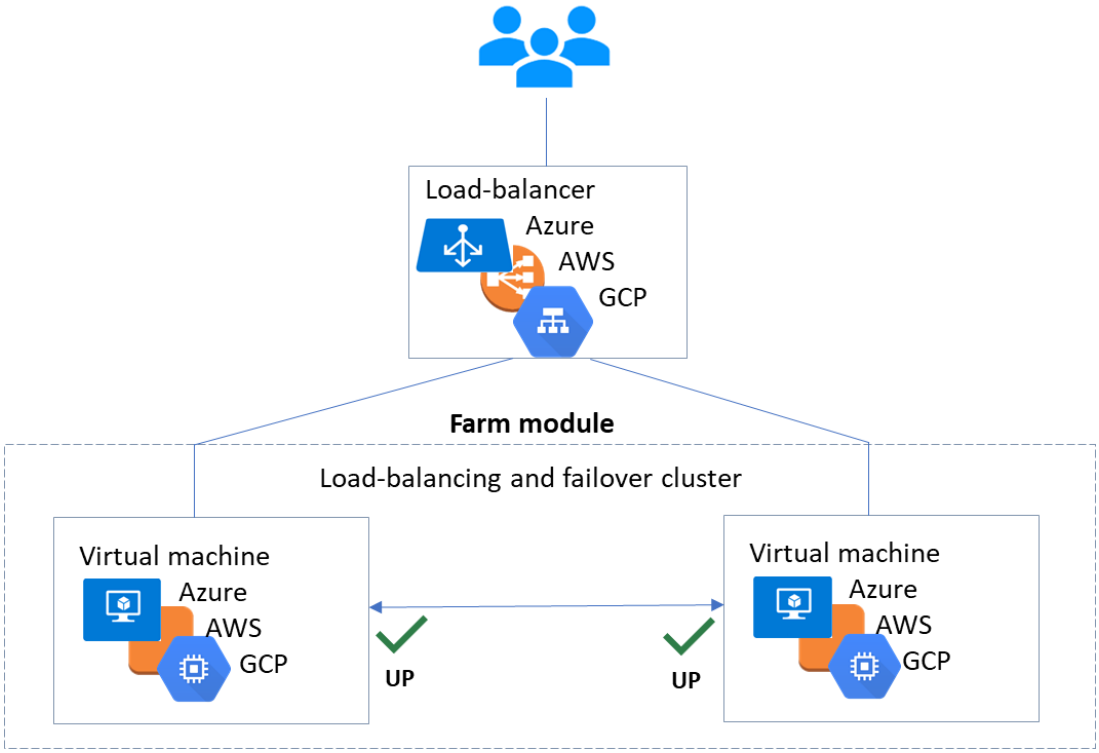
- l'application critique s'exécute sur le serveur `PRIM`
- les utilisateurs sont connectés à une adresse IP virtuelle principale/secondaire configurée dans le load balancer du cloud
- SafeKit implémente un vérificateur d'état de module générique que teste le load balancer. Sur le serveur `PRIM`, le vérificateur d'état renvoie OK et NOK sur le serveur `SECOND`
- sur chaque serveur, SafeKit surveille l'application critique à l'aide du détecteur de mort de processus et de checkers personnalisés

- SafeKit redémarre automatiquement l'application critique en cas de défaillance logicielle ou matérielle grâce à des scripts de redémarrage
- SafeKit effectue la réplication en temps réel synchrone de fichiers contenant des données critiques
- un connecteur à la console Web SafeKit est installé sur chaque serveur. Ainsi, le cluster à haute disponibilité peut être géré de manière très simple pour éviter les erreurs humaines

1.6.2 Cluster ferme dans Microsoft Azure, Amazon AWS et Google GCP

SafeKit est la plus simple solution dans les clouds Azure, AWS et GCP, pour mettre en œuvre un cluster actif-actif avec répartition de charge et failover applicatif (module ferme).

Pour un mise en œuvre rapide, se référer à cluster ferme dans Azure, cluster ferme dans AWS ou cluster ferme dans GCP.



- l'application critique s'exécute sur tous les serveurs `UP`
- les utilisateurs sont connectés à une adresse IP virtuelle, avec partage de charge, configurée dans le load balancer du cloud
- SafeKit implémente un vérificateur d'état de module générique que teste le load balancer. Le vérificateur d'état renvoie OK quand le serveur est `UP` ; NOK dans les autres cas (y compris en cas de panne matériel), ce qui arrête le routage du trafic vers ce serveur par le load balancer
- sur chaque serveur, SafeKit surveille l'application critique à l'aide du détecteur de mort de processus et de checkers personnalisés
- SafeKit redémarre automatiquement l'application critique en cas de défaillance logicielle ou matérielle grâce à des scripts de redémarrage

- un connecteur à la console Web SafeKit est installé sur chaque serveur. Ainsi, le cluster à haute disponibilité peut être géré de manière très simple pour éviter les erreurs humaines

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

2.1.3.1 Sur Windows en tant qu'Administrateur

2.1.3.1.1 Installation du package SafeKit

1. Se loguer en tant qu'administrateur
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.

Il est aussi possible d'installer le .msi en mode non interactif en exécutant dans un terminal PowerShell : `msiexec /qn /i safekitwindows_8_2_x_y.msi`

2.1.3.1.2 Setup du pare-feu

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

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 `.\private\bin\firewallcfg.cmd add`

Cela configure le pare-feu Microsoft pour SafeKit. Pour plus de détails ou d'autres pare-feu, voir la section 10.3 page 160

2.1.3.1.3 Initialisation du service web SafeKit

Cette étape est obligatoire pour initialiser la configuration par défaut du service web, qui est utilisé par la console web et la commande `safekit` globale. Par défaut, il est en effet nécessaire de s'authentifier pour accéder au service. Le script suivant facilite sa mise en œuvre en l'initialisant avec l'utilisateur `admin` et le mot de passe donné `pwd`, par exemple.

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 `.\private\bin\webservercfg -passwd 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 11.2.1 page 179.

Important

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.

Note

Sur upgrade, cette étape peut être ignorée si cela a déjà été fait lors de l'installation précédente de SafeKit 8.2. Si elle est réappliquée, cela aura pour effet de réinitialiser le mot de passe avec la nouvelle valeur.

2.1.3.2 Sur Linux en tant que root

2.1.3.2.1 Installation du package SafeKit

1. Se loguer en tant que root
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 page 160.

   - ✓ 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 11.2.1 page 179.

     Important

     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`)

2.1.3.2.2 Setup 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. Sinon, voir la section 10.3 page 160.

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. Sinon, voir la section 11.2.1 page 179.

2.1.4 Utilisation de la console 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 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 page 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 page 37.

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 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>`

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 une description complète de la commande, voir 9 page 143.

2.1.5 Clés de licence SafeKit

- Si vous n'installez pas de clé, le produit s'arrêtera tous les 3 jours
- Vous pouvez obtenir une clé d'essai d'un mois gratuit (fonctionne avec n'importe quel hostname/n'importe quel OS) : http://www.evidian.com/safekit/requestevalkey.php
- Pour obtenir des clés permanentes basées sur le nom de la machine/OS (voir section 8.2 page 134)
- Sauvegarder la clé de préférence dans le fichier `SAFE/conf/license.txt` (ou n'importe quel fichier dans `SAFE/conf`) sur chaque serveur
- Si des fichiers dans `SAFE/conf` contiennent plusieurs clés, la clé la plus favorable sera choisie.
- Vérifier la conformité de la clé avec la commande `SAFE/safekit level`

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 section 10.4 page 165)
- 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

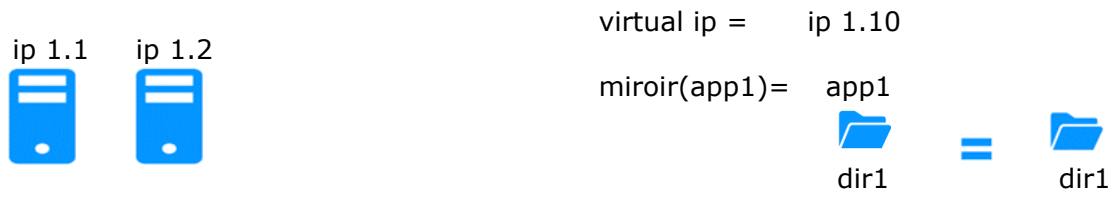
- En Linux, le package SafeKit dépend d'autres packages système. La plupart sont installés automatiquement, excepté ceux spécifiques à la mise en œuvre du load-balancing dans une ferme et de la réplication de fichiers dans un miroir.

  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 ainsi que le package `devel` correspondant à la version du kernel installé (`kernel-devel`).
- 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 (voir section 4.3.4 page 80)

- 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 Operating System
- 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 page 219

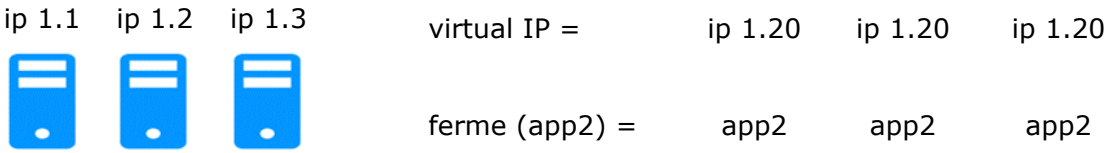
2.2.3 Prérequis application

- 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 page 29

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 page 219
- voir aussi la section 2.1.6 page 29

2.3.3 Prérequis application

Les mêmes prérequis que pour un module miroir décrits en 2.2.3 page 30.

2.4 Upgrade de SafeKit

2.4.1 Quand procéder à un upgrade ?

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.

2.4.2 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 ;`

Important

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)

2.4.3 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 `safekit uninstall`

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

   Voir section 10.3 page 160

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

2.4.4 Procédure de réinstallation et post-installation

1. Installer le nouveau package comme décrit en 2.1 page 25

2. Vérifier avec la commande `safekit level` la version SafeKit installée et la validité de la licence qui n'a pas été désinstallée

   Si vous avez un problème avec le nouveau package et l'ancienne clé, prendre une licence temporaire (voir section 2.1.5 page 29)

3. Si la console web est utilisée, vider le cache du navigateur web et forcer l'actualisation des pages HTML

4. Reconfigurer chaque module installé 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

5. Reconfigurer le démarrage automatique du module au boot si nécessaire

   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)

6. 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 :

➪ 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 page 177.

  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.

➪ 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 10.8 page 172.

2.5 Désinstallation complète de SafeKit

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

2.5.1 Sur Windows en tant qu'Administrateur

1. Arrêter tous les modules à l'aide de la commande `safekit shutdown`
2. 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:`)
3. Désinstaller le package SafeKit via Control Panel-Add/Remove Programs
4. Redémarrer le serveur
5. Détruire le répertoire `SAFE` qui correspond à l'installation précédente de SafeKit
6. Défaire les modifications effectuées pour configurer le démarrage au boot/l'arrêt au shutdown de SafeKit

   Voir la section 10.4 page 165
7. Défaire les modifications manuelles effectuées sur le pare-feu

   Voir section 10.3 page 160

2.5.2 Sur Linux en tant que root

1. Arrêter tous les modules à l'aide de la commande `safekit shutdown`
2. Fermer tous les éditeurs, explorateur de fichiers, ou terminaux sous `SAFE` et `SAFEVAR` (`SAFE=/opt/safekit` ; `SAFEVAR=/var/safekit`)
3. 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
4. Redémarrer le serveur

5. Défaire les modifications effectuées pour paramétrer les règles de pare-feu

   Voir section 10.3 page 160

6. Supprimer, l'utilisateur/groupe créés par l'installation précédente (par défaut `safekit/safekit`) avec les commandes :

   ```
   userdel safekit
   groupdel safekit
   ```

2.6 Documentation produit

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 section 8 page 133.
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 ⋮/⓪ 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.

Note

Consulter le Release Notes, sur `https://support.evidian.com/safekit`, pour la liste des restrictions et problèmes connus avec la console web.

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. La console web fonctionne également sur des mobiles et tablettes. Consulter le
- 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

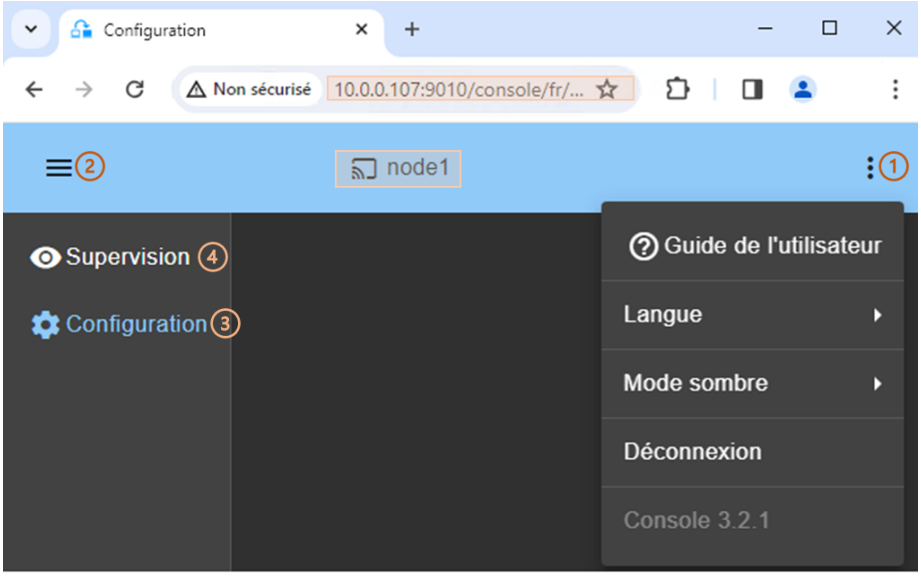
Par défaut, l'accès à la console web nécessite que l'utilisateur s'authentifie avec un nom et mot de passe. A l'installation de SafeKit, vous avez dû l'initialiser avec l'utilisateur `admin` et lui affecter un mot de passe. Ce nom `admin`, et ce mot de passe sont suffisants pour accéder à toutes les fonctionnalités de la console. Pour plus de détails sur cette configuration, voir 11.2.1 page 179.

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`.

   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.

   Note: 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.

3. 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`).
4. 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.

Note

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 page 205). Pour que le fonctionnement soit correct, il est impératif que la configuration du cluster soit identique sur tous les nœuds.

Important

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

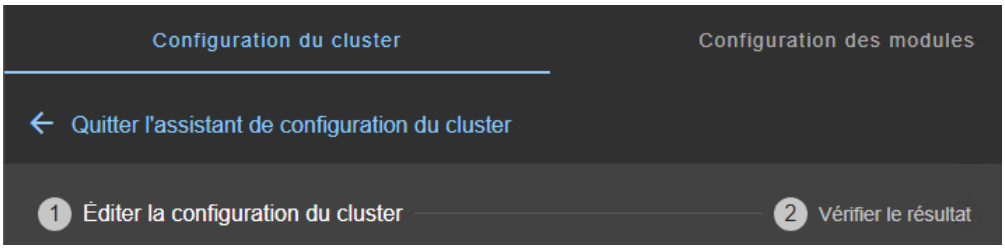
Ouvrir l'assistant de configuration :

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

Ou

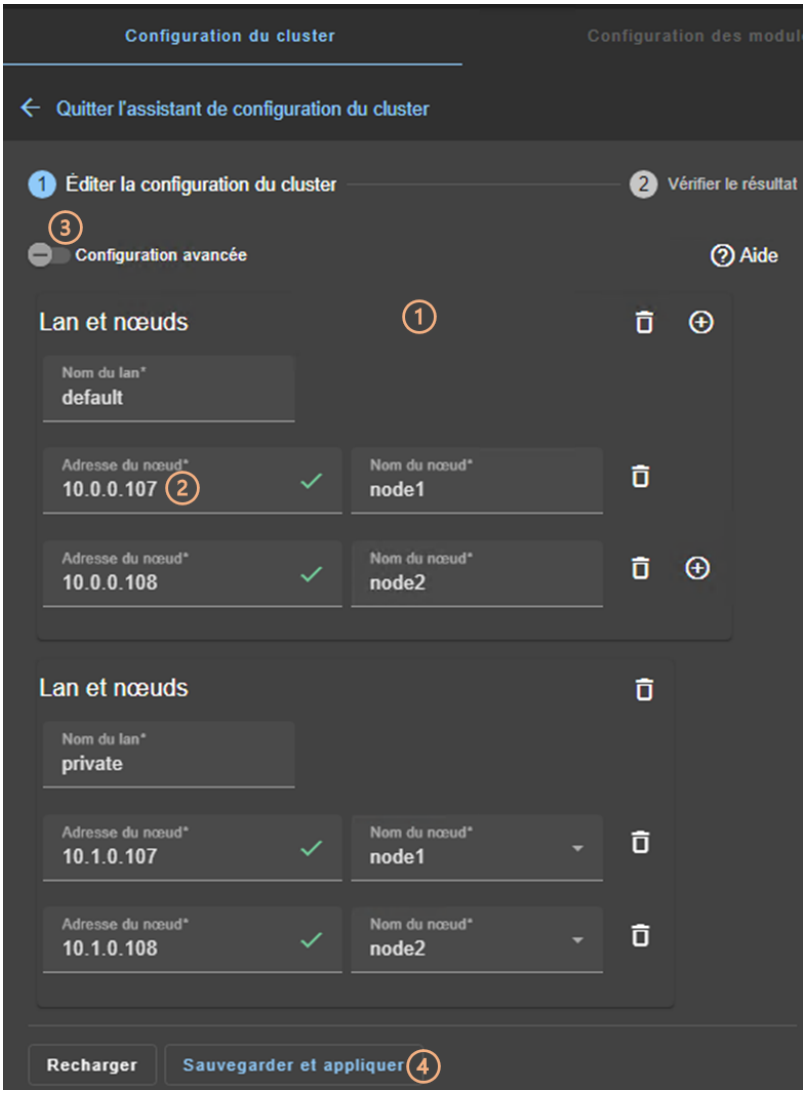
- ✓ en naviguant dans la console via ⚙Configuration/Configuration du cluster/ ⚙Configurer le cluster

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



1. Éditer la configuration du cluster décrit page 41
2. Vérifier le résultat décrit page 43
3. ← pour Quitter l'assistant de configuration du cluster

3.2.1.1 Éditer la configuration du cluster



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

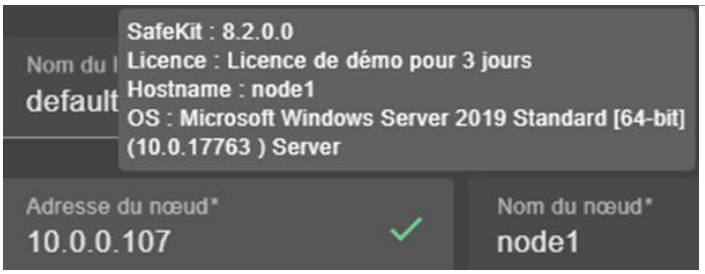
  Cliquer sur ⊕ pour ajouter un nouveau nœud /lan ou sur 🗑 pour supprimer un nœud/lan du cluster.

  Important

  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 section 7.1 page 111.

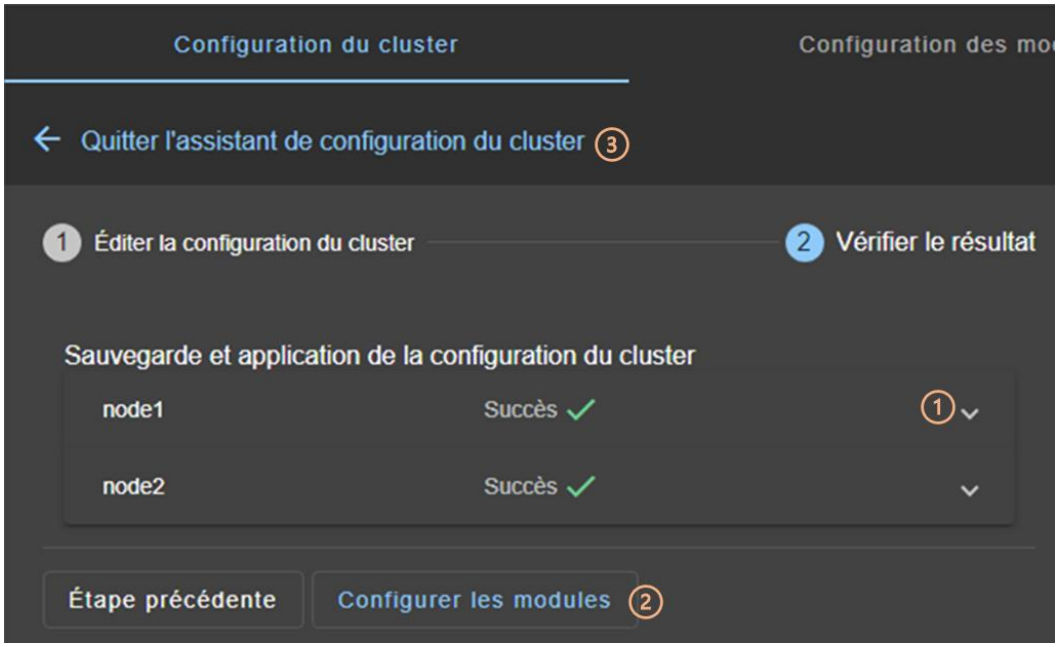
- Modifier le nom du nœud si besoin. Ce nom est celui qui sera utilisé par le service d'administration de SafeKit pour identifier de manière unique chaque nœud du cluster. C'est également le nom affiché dans la console web.
- (3) Si besoin, cliquer sur Configuration avancée pour basculer sur l'édition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

Important

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

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

- 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 configuration du cluster

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

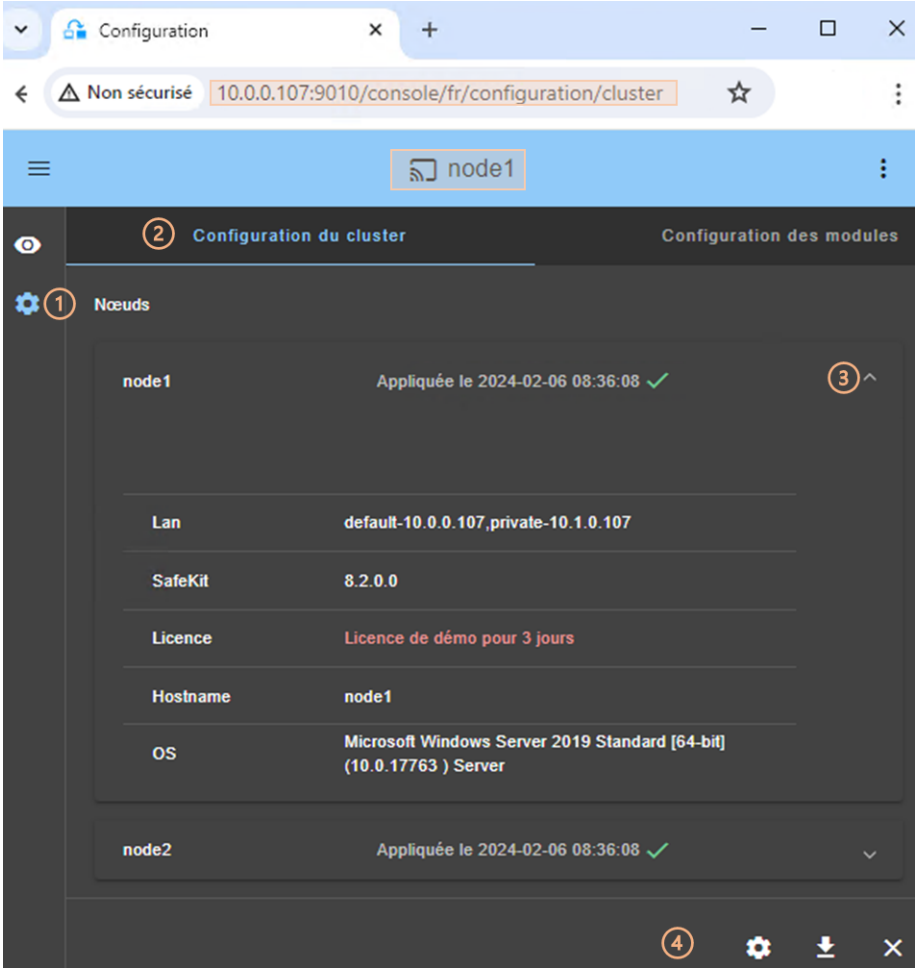
L'ouvrir :

✓ 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 (nom des lans et adresses définies dans la configuration du cluster…)
- (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

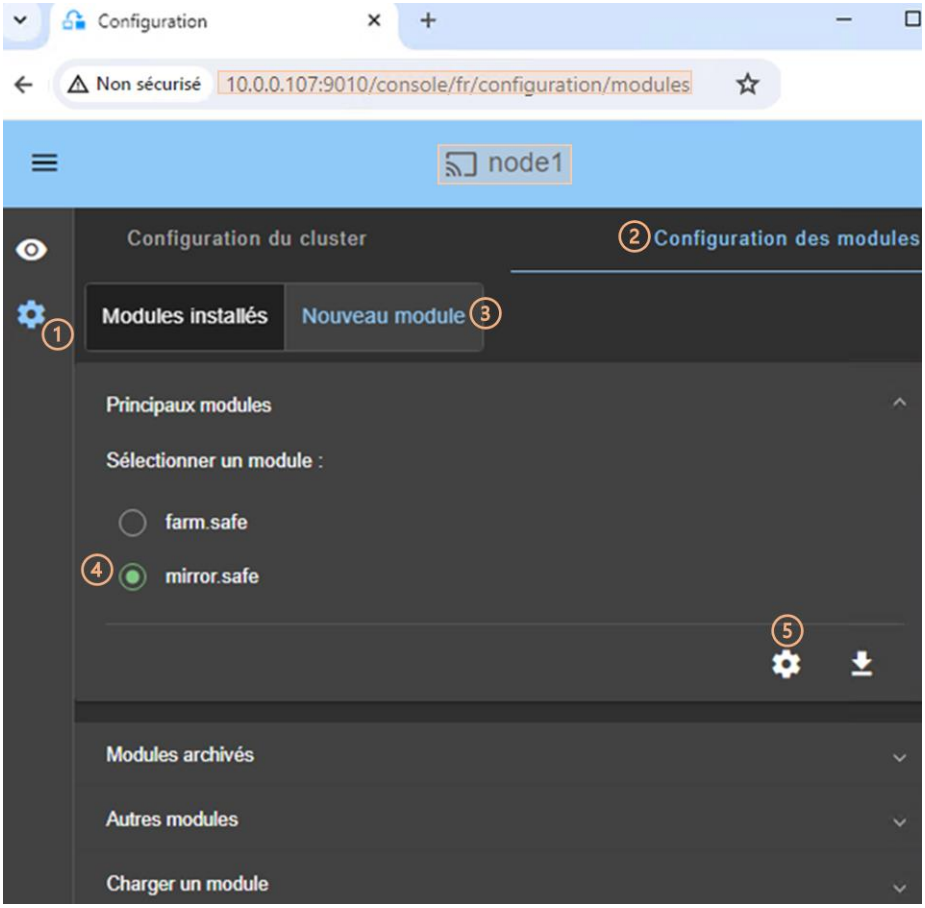
Ou

✓ 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.

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` et `farm.safe` 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.

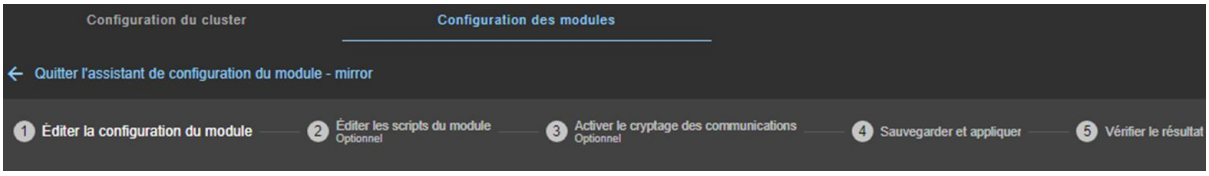
  Ils sont récupérés depuis `node1` sous `SAFE/Application_Modules/other`.
- ✓ Un module stocké localement accessible depuis Charger un module.

- (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 page 47
2. Éditer les scripts du module (Optionnel) décrit page 48
3. Activer le cryptage des communications (Optionnel) décrit page 49
4. Sauvegarder et appliquer décrit page 50
5. Vérifier le résultat décrit page 51

6. ← pour Quitter l'assistant de configuration du module

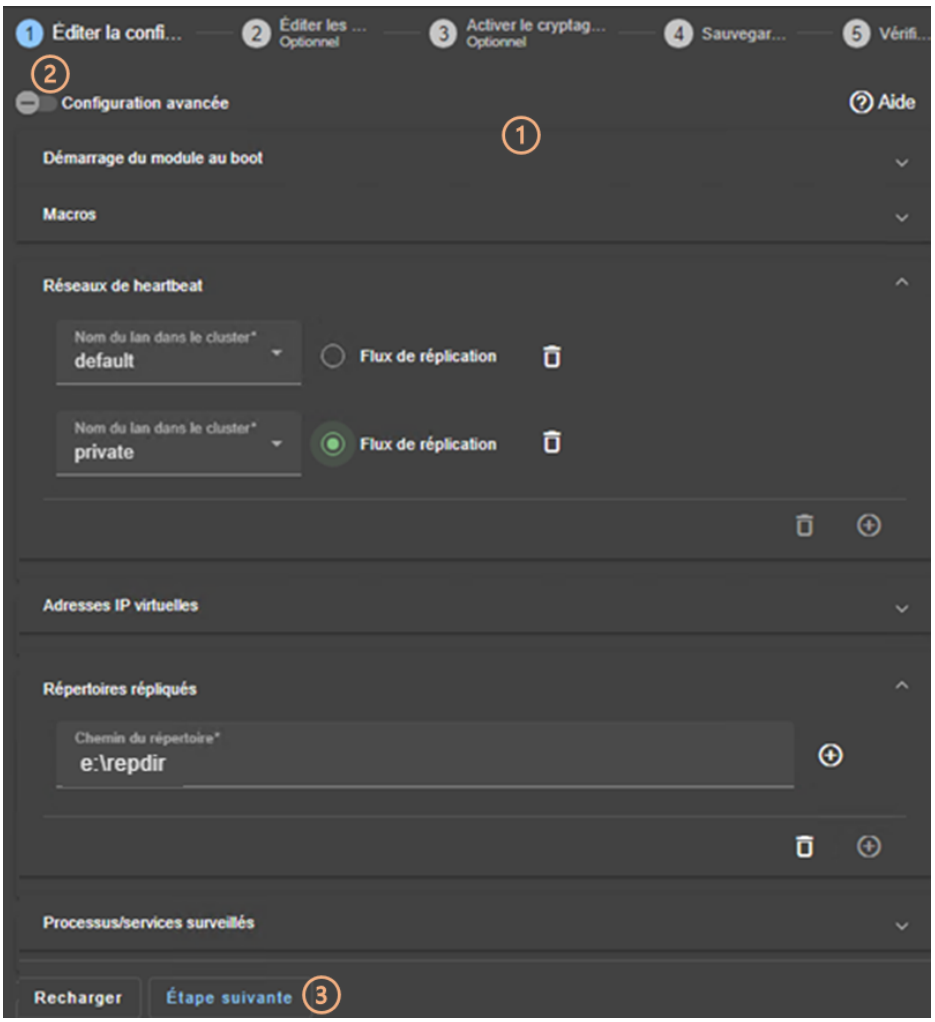
Notez que la reconfiguration d'un module ne peut être appliquée qu'aux nœuds sur lesquels le module en question n'est pas démarré. Il convient donc d'arrêter le module avant de lancer l'assistant de configuration.

Important

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.

Note

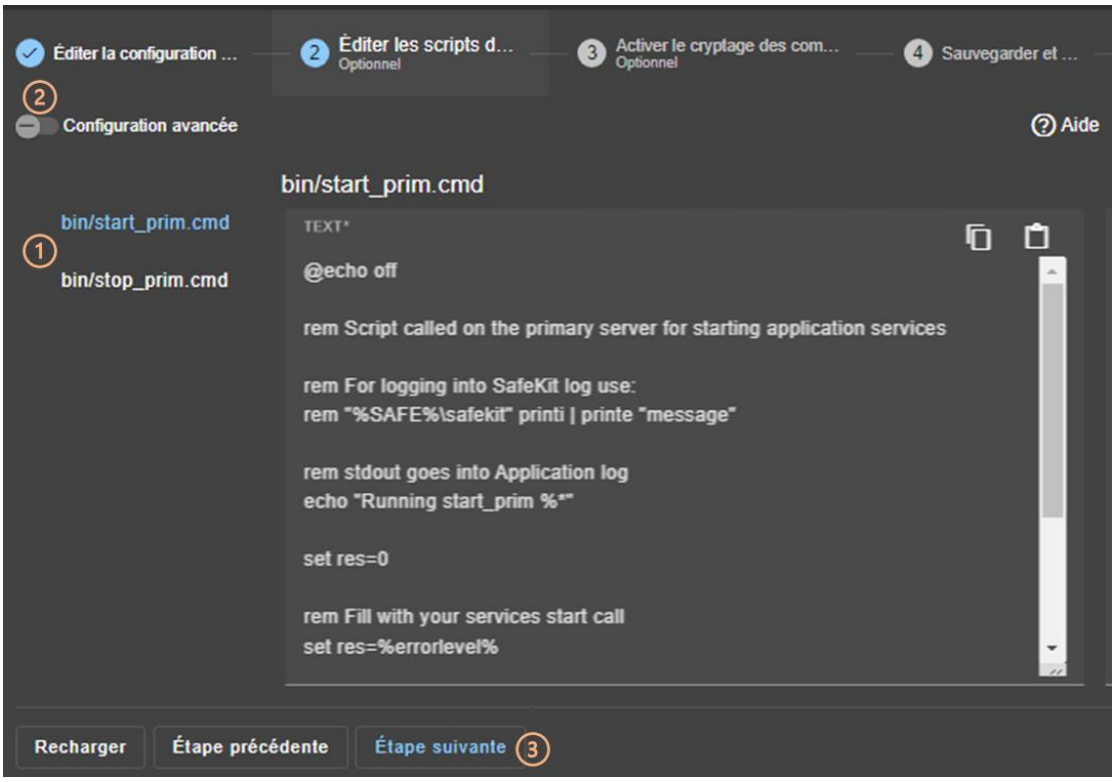
Le nom des Réseaux de heartbeat proposés sont les noms des lans saisis lors de la configuration du cluster.

- (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.

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.

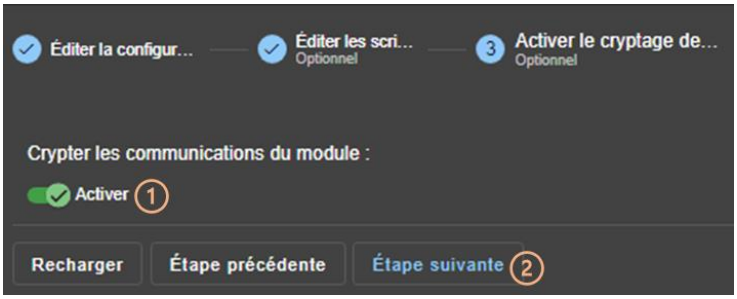
  Cliquer sur ⧉ pour copier le contenu du script et l'éditer avec votre éditeur syntaxique favori. Une fois fait, copier le contenu mis à jour dans le champ d'input avec 📋.
- (2) Si besoin, cliquer sur Configuration Avancée pour lister les autres scripts du module et les éditer (`prestart`, `poststop`,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

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 section 10.5 page 166.



- (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.

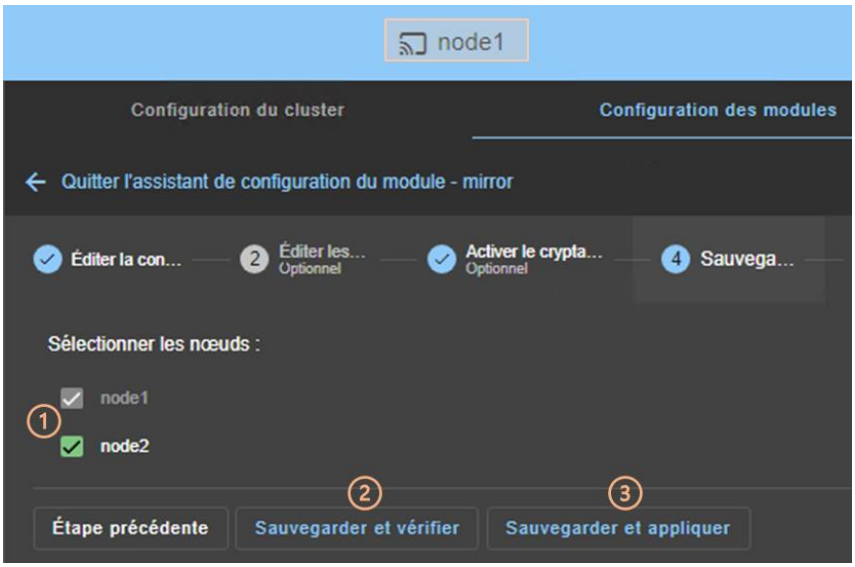
Pour générer de nouvelles clés de cryptage, il faut :

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.

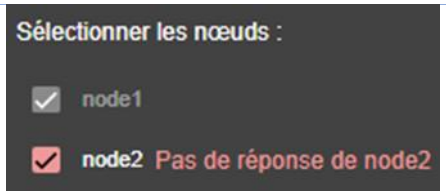


- (1) Cochez/décochez pour sélectionner/désélectionner les nœuds. Veuillez noter que le nœud de connexion (`node1` dans l'exemple) est obligatoire.

  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 section 7.1 page 111.

  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.

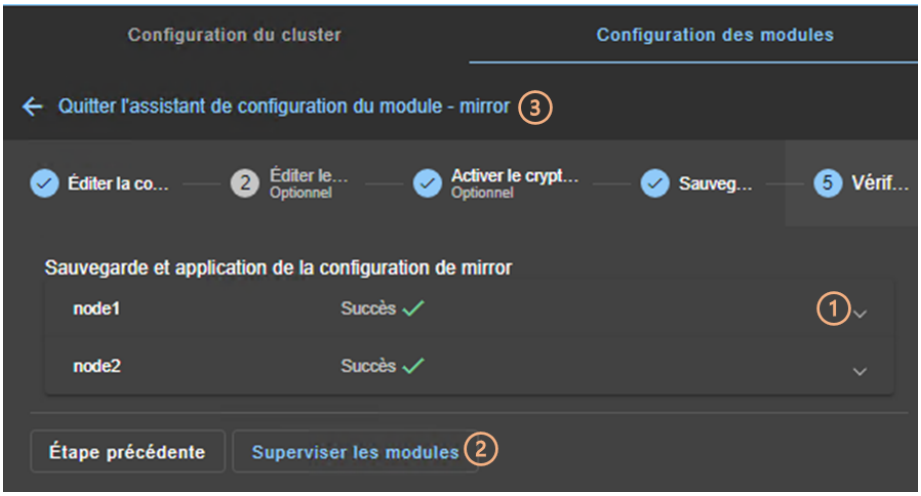
  Si l'opération est réussie, la configuration appliquée devient la configuration active du module.

Important

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 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.

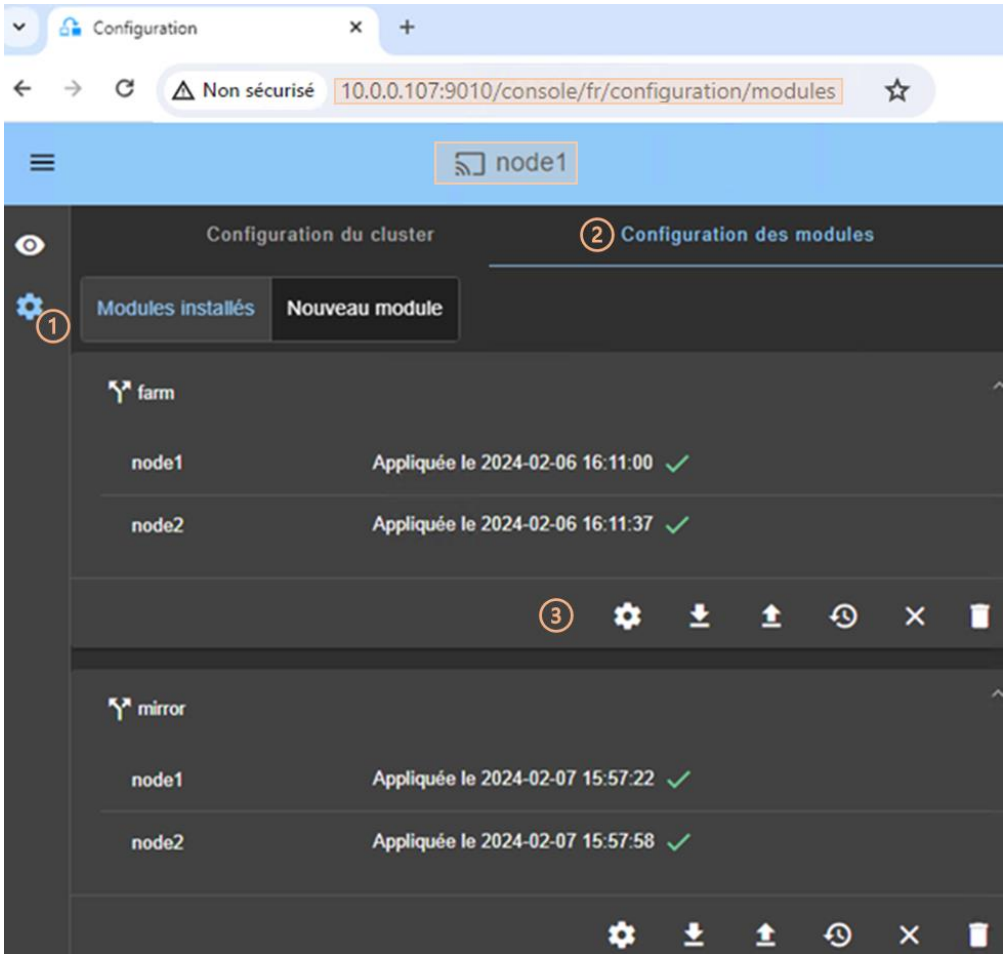
L'ouvrir :

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

Ou

✓ En naviguant dans la console sur ⚙Configuration/Configuration des modules

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 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.
- ⬇ pour télécharger le `.safe`, composé de tous les fichiers du module (`userconfig.xml`, scripts) depuis le nœud de connexion.
- ⬆ pour reconfigurer le module depuis le contenu d'un `.safe` stocké localement.
- ⟲ 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).
- ✕ pour déconfigurer le module sur un ou plusieurs nœuds, sans le désinstaller. Ses fichiers de configuration sont conservés pour être éventuellement réappliquer plus tard.
- 🗑 pour désinstaller complètement le module sur un ou plusieurs nœuds.

  L'ensemble des fichiers de configuration du module sont empaquetés dans un fichier `.safe` qui est archivé du côté serveur sous `SAFE/Application_Modules/backup`.

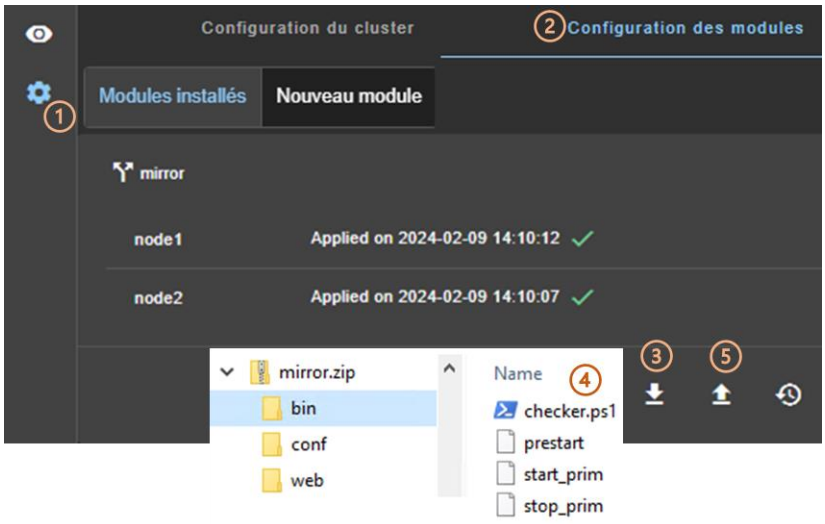
Important Avant chaque reconfiguration, déconfiguration et désinstallation, sur chaque nœud, fermer tous les éditeurs, explorateur de fichiers, shells ou cmd sous `SAFE/modules/AM` (au risque sinon que l'opération échoue).

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

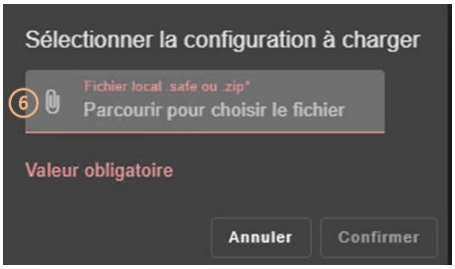
3.3.4 Ajouter un script au module

Il se peut que vous deviez ajouter des scripts au module, tels que des custom checkers, à votre configuration actuelle du module.

Dans cet exemple, le script `checker.ps1` est ajouté au module `mirror`.



- (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) Éditer le `mirror.safe` (qui est un fichier zip) pour ajouter vos scripts sous le répertoire bin (`checker.ps1` dans l'exemple).
- (5) Charger le `mirror.safe` modifié (l'extension zip est aussi acceptée).



- (6) Cliquer sur pour sélectionner le fichier à charger, puis Confirmer.
- L'assistant de configuration du module est lancé avec le contenu de ce fichier. Les nouveaux scripts sont visibles avec la Configuration avancée à l'étape 2. 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…).

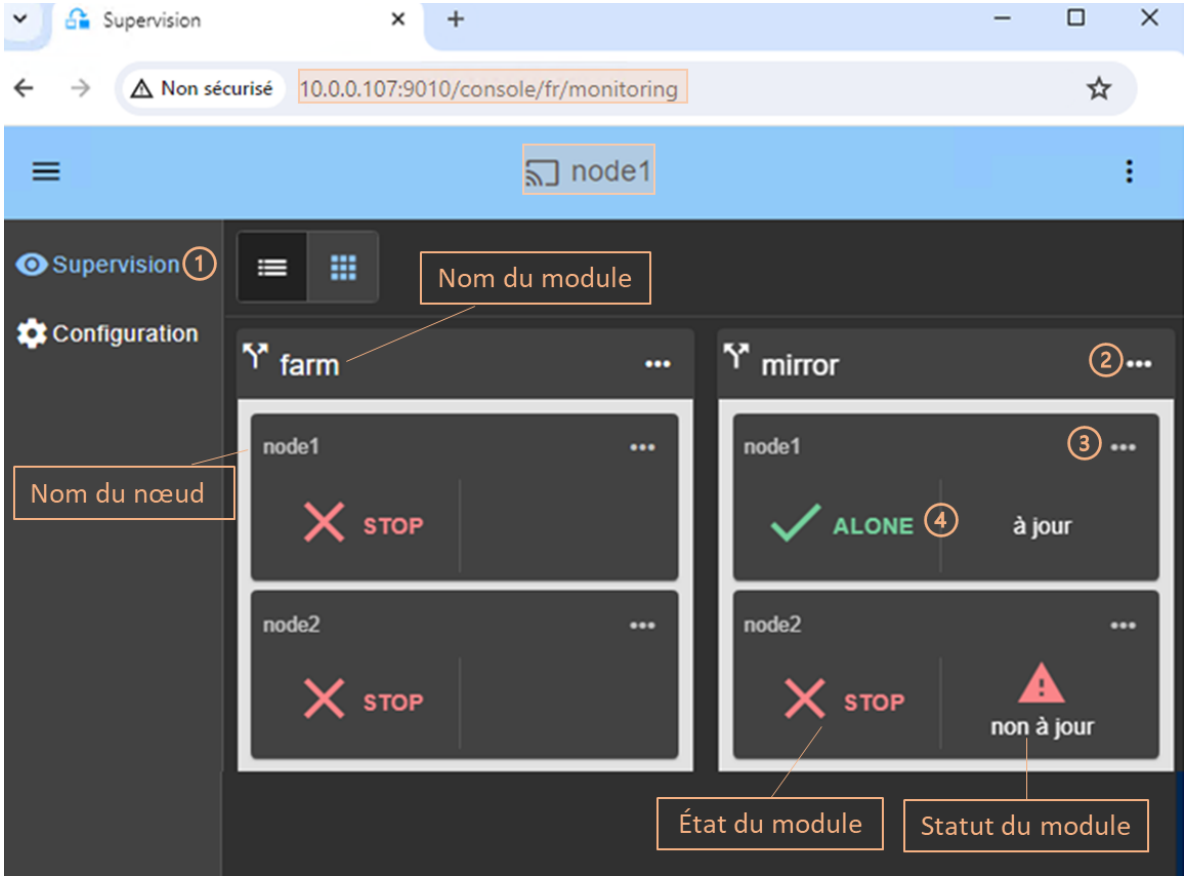
La page d'accueil de supervision des modules est accessible :

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

Ou

✓ En naviguant dans la console sur Supervision

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 et son status sur chaque nœud

  Pour sa description, voir 3.4.1 page 56.
- (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).

  (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 leur description, voir 3.4.2 page 57.
- (4) Cliquer sur le panneau du nœud (`mirror>node1` dans l'exemple) pour ouvrir les détails du module sur ce nœud (journaux, ressources…).

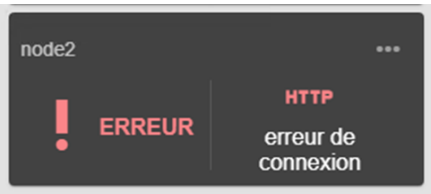
  Pour sa description, voir 3.4.3 page 59.

3.4.1 État et status d'un module

➪ L'état d'un module sur un nœud est l'un des états suivants.

Le module est installé mais non configuré :

Le nœud ne répond pas 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 section 7.1 page 111.

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 SafeKit.

Le module est configuré et le nœud répond :

`STOP`	arrêté (prêt à démarrer)
`WAIT`	en attente d'une ressource
`ALONE`	primaire sans secondaire (module miroir)
`PRIM`	primaire avec secondaire (module miroir)
`SECOND`	secondaire avec primaire (module miroir)
`UP`	actif (module ferme)

Avec les icônes/couleurs associés qui signifient :

✕ ou ○ `NotReady`	état bloqué
`Transient`	état transitoire
✓ `Ready`	état stable

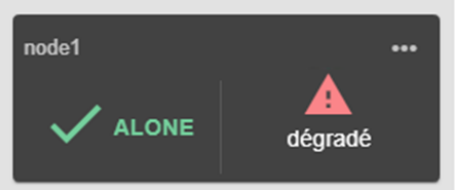
Pour la description des changements d'états d'un module miroir, voir section 5.2 page 97.

Pour la description des changements d'états d'un module miroir, voir section 6.2 page 108.

➪ Le status du module est l'un des suivants.

Pour un module miroir, il affiche le status des répertoires répliqués : `à jour` ou `non à jour`.

Dans le cas particulier du mode dégradé (voir 7.6 page 117), il affiche :



Pour un module ferme, il affiche la part de partage de charge réseau sur l'IP virtuelle : `0%`, `50%` or `100%` (pour 2 nœuds).

Lorsque le module (ferme ou miroir) est dans l'état `WAIT (NotReady)`, la raison est affichée, généralement le nom de la règle de failover qui bloque le module jusqu'à ce que la ressource associée repasse de l'état `down` à l'état `up`. Pour plus de détails, voir 7.9 page 118.

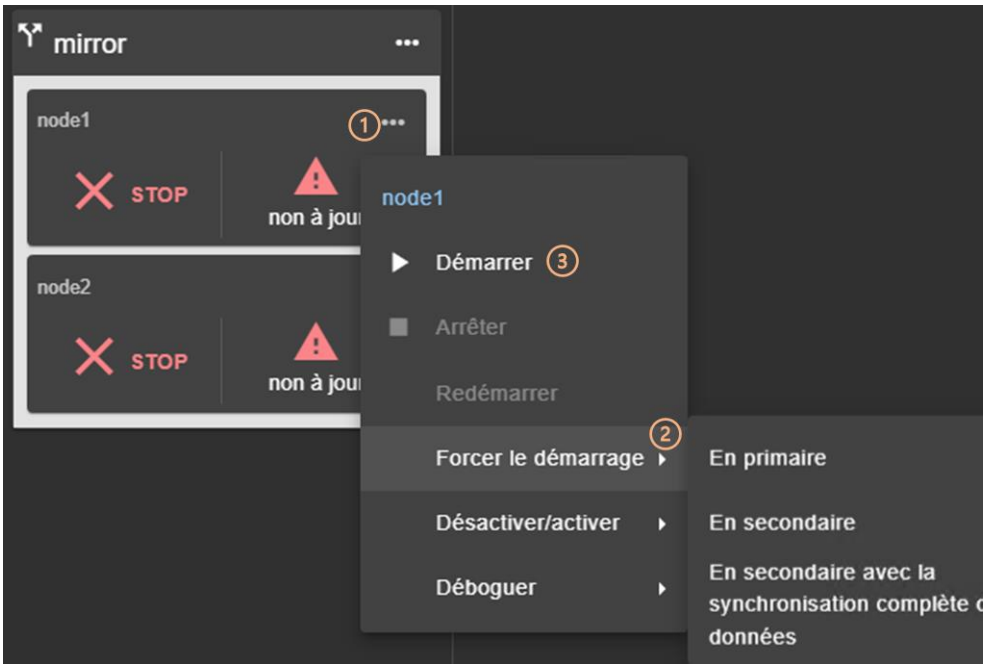
Dans l'exemple ci-dessus, le module est bloqué par la règle de failover nommée `c_checkfile`. Pour analyser le problème, lisez les journaux et les états des ressources comme décrit plus loin.

Quand le nœud ne répond pas, le status est erreur de connexion.

3.4.2 Menus de contrôle d'un module

➪ Contrôler un module miroir

Dans cet exemple, le module `mirror` est configuré sur `node1` et `node2`.



- (1) Cliquer sur ••• pour ouvrir le menu d'actions sur `node1`.
- (2) Utiliser Forcer le démarrage lorsque vous devez décider quel nœud doit démarrer en primaire ou secondaire.

  Par exemple, au 1er démarrage d'un module miroir, vous devez Forcer le démarrage /En primaire le nœud qui a les répertoires répliqués à jour.
- (3) pour les démarrages suivants, cliquer sur ▷ Démarrer, car SafeKit mémorise le dernier nœud à jour.
- Cliquer sur Debug pour télécharger les journaux ou snapshots du module depuis un seul nœud, ou depuis tous les nœuds.

Se référer aux sections listées ci-dessous :

✓ Pour le premier démarrage d'un module miroir, voir section 5.3 page 98

✓ Pour le démarrage d'un module miroir avec les données à jour, voir section 5.5 page 100

✓ Pour continuer les tests, voir section 4 page 69.

✓ Pour comprendre et vérifier le bon fonctionnement d'un module miroir, voir section 5 page 95

➪ Contrôler un module ferme

Dans cet exemple, le module `farm` est configuré sur `node1` et `node2`.

- (1) Cliquer sur ••• pour ouvrir le menu d'actions globales.
- (2) Cliquer sur ▷ Start pour démarrer le module sur `node1` et `node2`.
- (3) Cliquer sur ••• pour ouvrir le menu et exécuter des actions uniquement sur `node2`.
- Cliquer sur Debug pour télécharger les journaux ou snapshots du module depuis un seul nœud, ou depuis tous les nœuds.

Se référer aux sections listées ci-dessous :

✓ Pour continuer les tests, voir section 4 page 69.

✓ Pour comprendre et vérifier le bon fonctionnement d'un module ferme, voir section 6 page 107

3.4.3 Détails du module

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

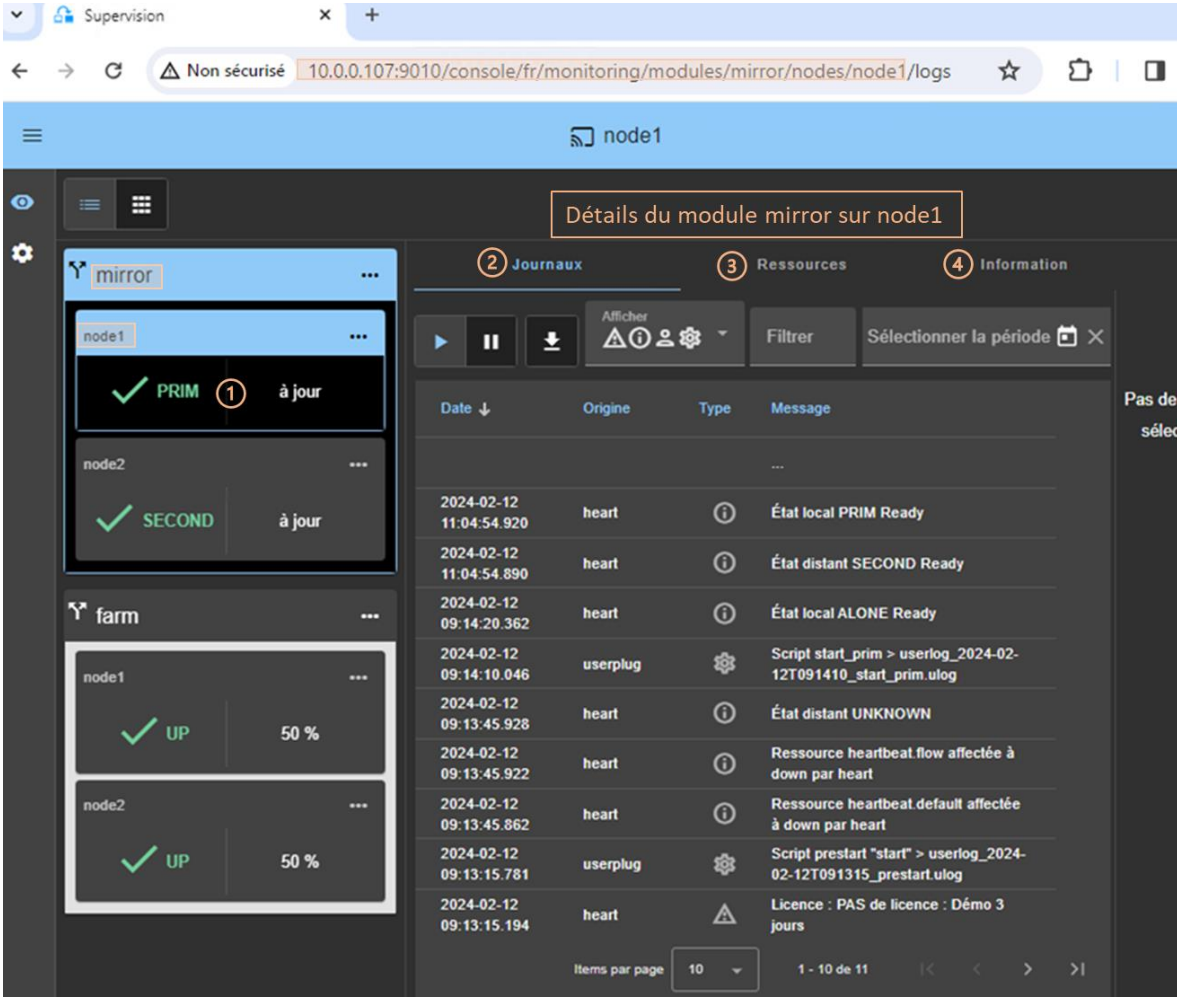
✓ Directement via l'URL http://host:9010/console/fr/monitoring/modules/AM/nodes/node (remplacer `node` par le nom du nœud et `AM` par le nom du module)

Ou

✓ En naviguant dans la console sur Supervision/Cliquer sur module>nœud

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

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



- (1) Cliquer sur le panneau du nœud (`mirror>node1` dans l'exemple) pour ouvrir 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 (SafeKit version and licence…).

3.4.3.1 Le journal du module et le journal des scripts

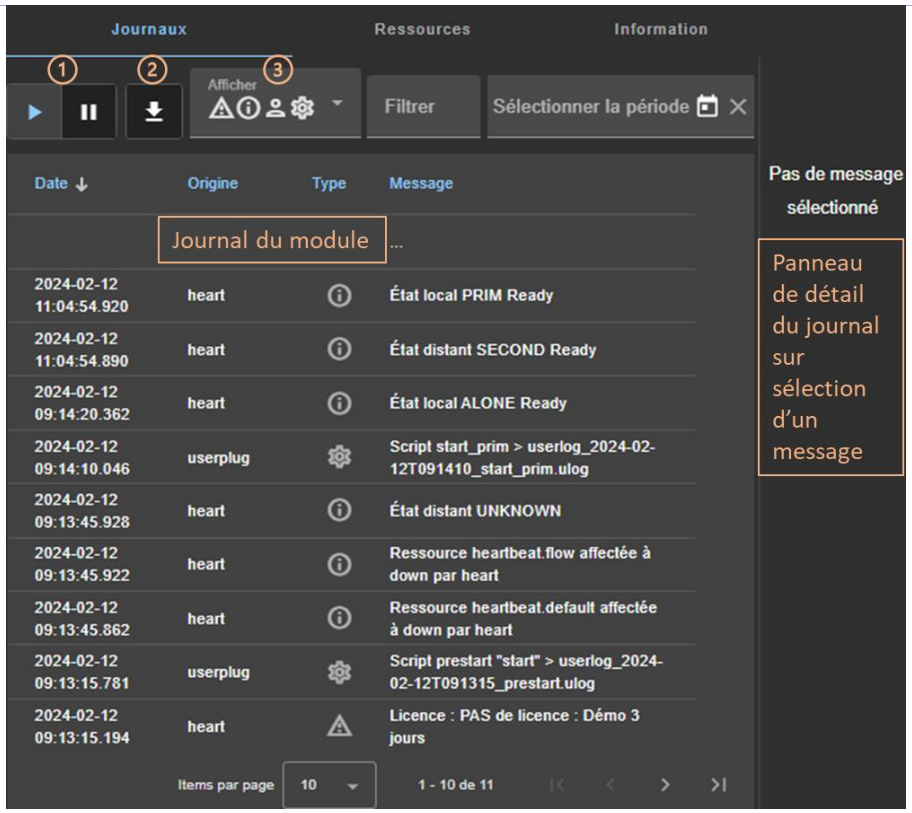
Vous pouvez afficher les journaux d'un module sur un nœud :

✓ 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)

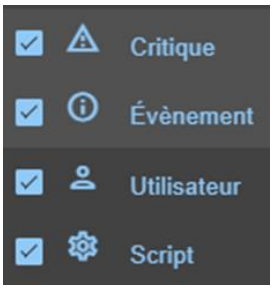
Ou

✓ En naviguant dans la console sur Supervision/Cliquer sur module>nœud/Onglet Journaux

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



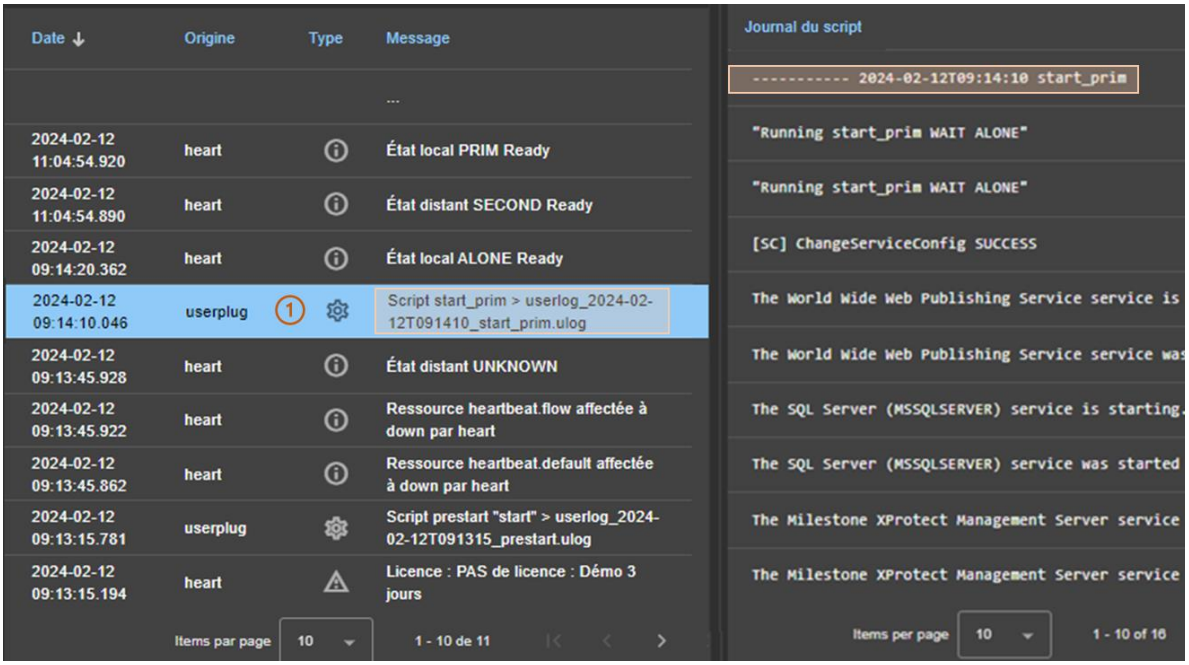
- (1) Cliquer sur ▶II pour reprendre/suspendre la visualisation en temps réel du journal du module.
- (2) Cliquer sur ⬇ pour télécharger le journal du module (non verbeux ou verbeux).
- (3) Sélectionner le type de messages à afficher :



- ⇨ 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

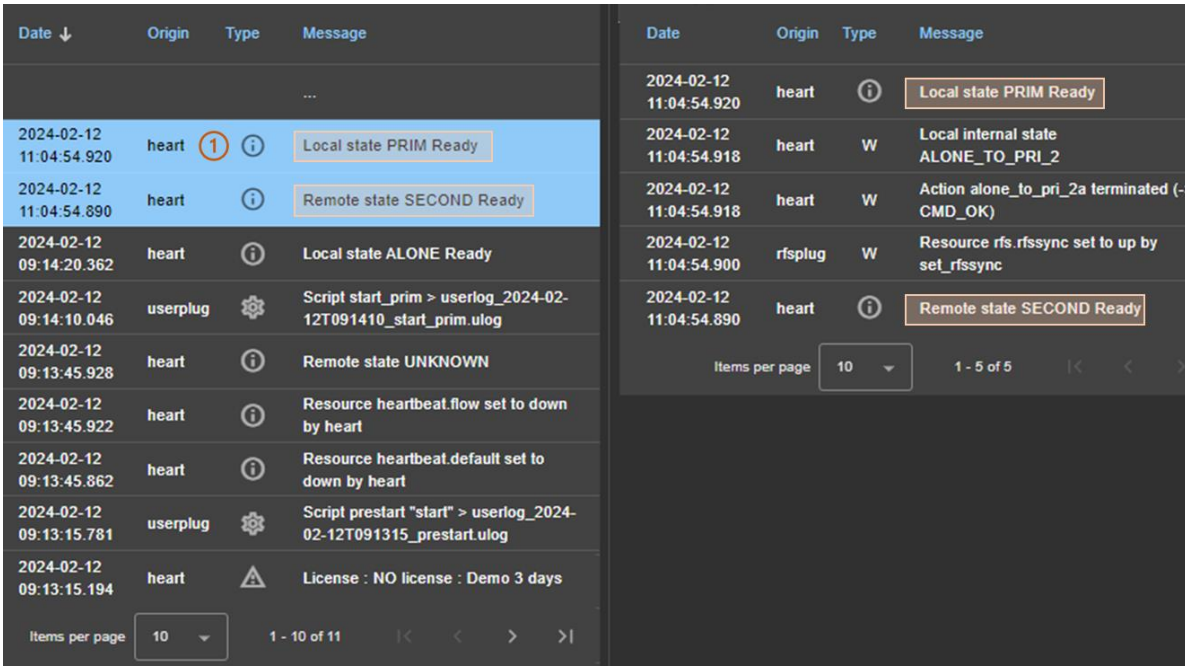
- Cliquer sur un message pour afficher le journal verbeux du module ou le journal des scripts (sortie des scripts) dans le détail du journal dans le panneau de droite.

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)

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.

Voir la section 7 page 111, pour une liste de messages démontrant un problème.

3.4.3.2 Les ressources du module

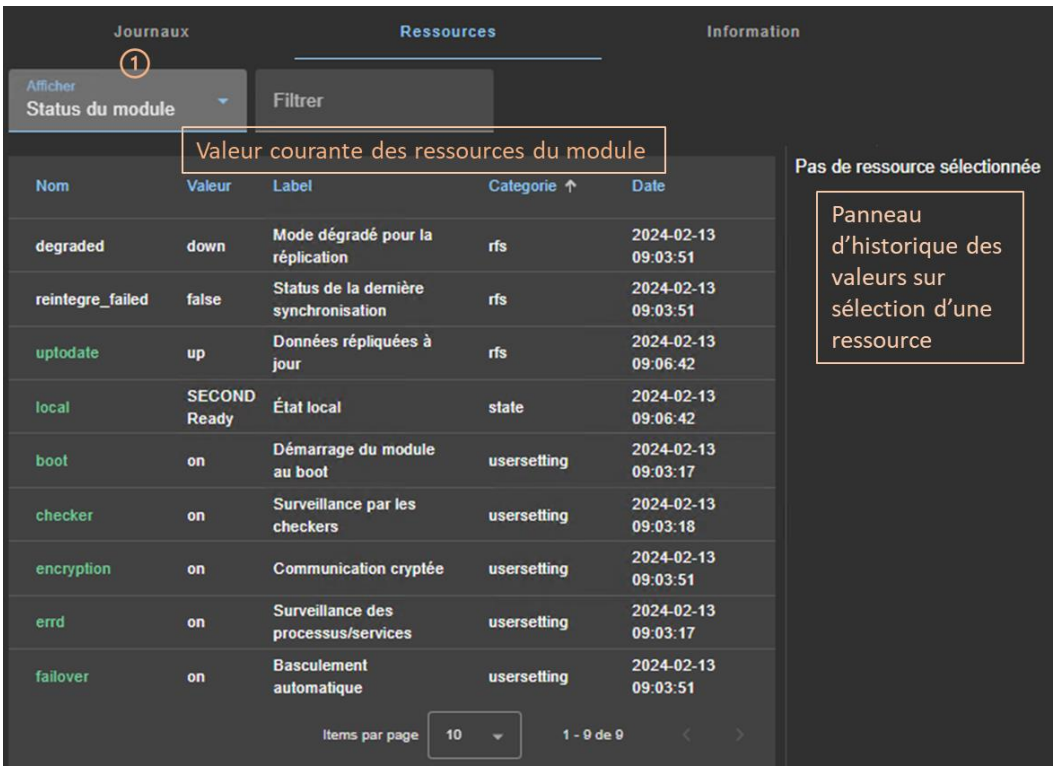
Vous pouvez afficher les ressources d'un module sur un nœud :

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

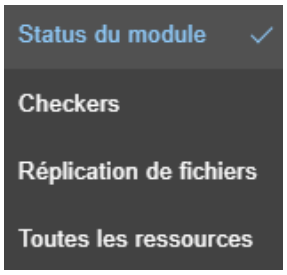
Ou

- ✓ En naviguant dans la console sur 👁 Supervision/Cliquer sur module>nœud/Onglet Ressources

Le panneau de gauche affiche 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

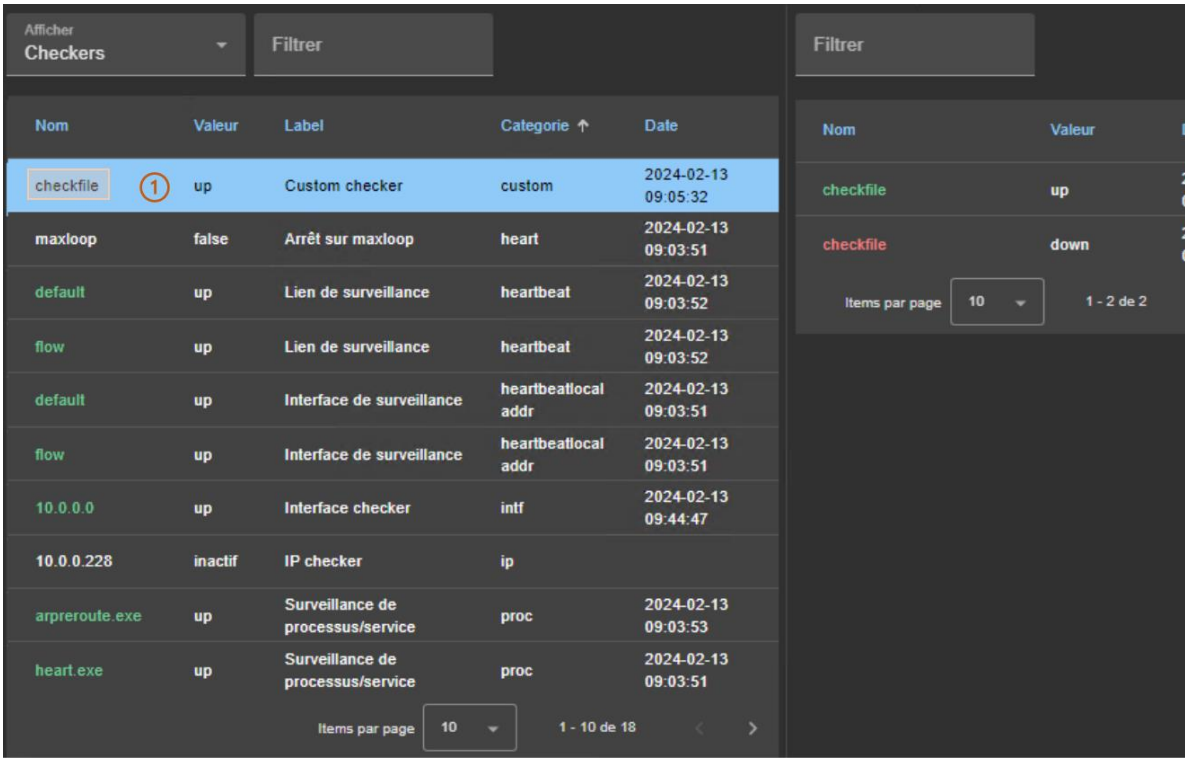
  Ressources affectées par des checkers
- ✓ Réplication de fichiers

  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 page 266).

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

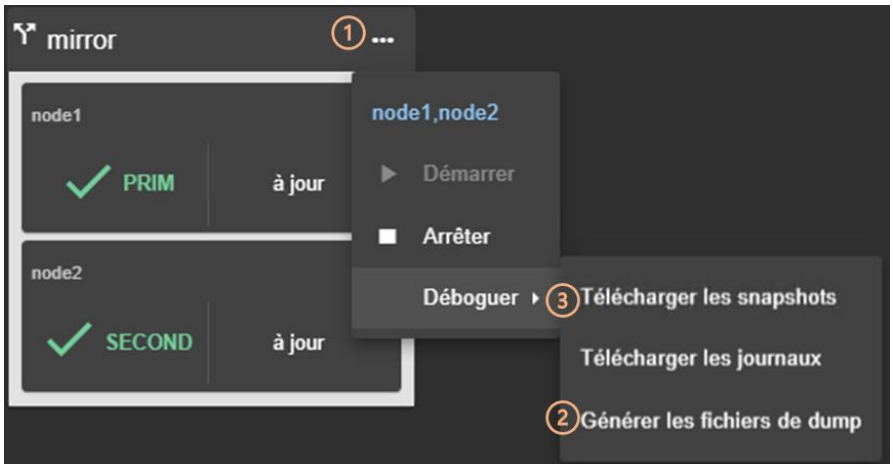


- (1) Cliquer sur la ligne qui consiste en :
  - ✓ la dernière date à laquelle la ressource a été affectée
  - ✓ le nom et la catégorie de la ressource. Le nom complet de la ressource est de la forme `<catégorie>.<nom>` (`custom.checkfile` dans l'exemple).
- L'historique des valeurs de la ressource est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

3.5 Snapshots d'un module pour le 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 page 124. Si cette analyse échoue, envoyez les snapshots au support comme décrit dans la section 8 page 133.

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 d'actions globales.
- (2) 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) 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`.

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 appliqué en HTTP ou HTTPS.

- HTTP/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. Elle peut être appliqué en HTTP ou HTTPS.

Pour les mettre en œuvre, se référer à la section 11 page 177.

4. Tests

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 page 115.

4.1 Installation et tests après boot

4.1.1 Test installation package

Installation du 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 :
  - in Windows

    `SAFE=C:\safekit` si `SystemDrive=C:`
    `SAFEVAR=C:\safekit\var`
  - in Linux
    `SAFE=/opt/safekit"`

    `SAFEVAR=/var/safekit`

  Pour plus d'informations, voir la section 10.1 page 157.
- 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 : 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 processus SafeKit après boot

Voir aussi la section 10.4 page 165

Test du service `safeadmin` :

- Le processus `safeadmin` doit apparaître dans la liste des processus
- Sans ce processus, aucune commande `safekit` n'est réalisable et rend :

  "Waiting for `safeadmin` ....."
  "Error: `safeadmin` administrator daemon not running"
- En Windows, `safeadmin` est un service et il se démarre dans l'interface Services de Windows

Test du service `safewebserver` :

- `safekit boot webstatus` retourne le démarrage ou non du service `safewebserver` au boot ("on" ou "off", "on" par défaut)
- Les processus httpd doivent apparaître dans la liste des processus
- Sans ces processus, la console web n'arrive pas à se connecter aux serveurs ; les checkers de <module> (`userconfig.xml`) et les commandes distribuées au cluster ne peuvent pas fonctionner
- Pour démarrer/arrêter le service `safewebserver`, `safekit webserver start |stop |restart`

Test du service `safeagent` (uniquement en Windows):

- `safekit boot snmpstatus` retourne le démarrage ou non du service `safeagent` au boot ("on" ou "off", "off" par défaut)
- Le processus `safeagent` doit apparaître dans la liste des processus
- Pour démarrer/arrêter le service `safeagent` taper, `safekit safeagent start |stop |restart`

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 page 159)
- `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
- aller dans `SAFE/modules/AM/conf` (où AM est le nom du module); le fichier `userconfig.xml` donne le type de module, mirror ou farm: `<service mode="mirror">` ou `<service mode="farm">`

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

- message dans les journaux du module sur les 2 serveurs (pour lire les journaux, voir 7.3 page 115)

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

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

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

- et dans le 2nd log

  "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`"

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

- message dans le journal du module (pour lire les journaux voir 7.3 page 115)

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

- le module arrêté exécute le script `stop_prim` qui arrête 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 :

  "End of stop"

  "Local state STOP NotReady"

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

  "Reason of failover: remote stop"

- l'application est redémarrée avec le script `start_prim` script du module qui passe dans l'état ALONE sur le serveur de reprise avec le message dans son journal :

  "Script `start_prim`"

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

- message dans le journal du module redémarré (pour lire les journaux voir 7.3 page 115)

  "Action start called by web@<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

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

- message dans le journal du module redémarré (pour lire les journaux voir 7.3 page 115)

  "Action restart called by web@<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)

4.2.5 Test swap du module miroir d'un serveur vers l'autre

- message dans le journal du module où la commande swap est passée (pour lire les journaux voir 7.3 page 115)

  "Action swap called by web@<IP>/SYSTEM/root"

  "Transition SWAP from SYSTEM"

  "Begin of Swap"

- Et dans le journal du module sur l'autre serveur, seulement :

  "Begin of Swap"

- inverse les rôles de PRIM et SECOND entre les 2 serveurs
- le script stop_prim est d'abord exécuté sur l'ancien module PRIM avec dans son journal :

  "Script stop_prim"

- puis le script start_prim est exécuté sur le nouveau module PRIM avec dans son journal :

  "Script start_prim"

- à la fin du swap, le module ✓PRIM (Ready) et le module ✓SECOND (Ready) sont inversés sur les 2 serveurs et l'application s'exécute sur le module PRIM

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>
 <interface arpreroute="on">
 <real_interface>
   <virtual_addr addr="virtip"
             where="one_side_alias"/>
  </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 swap -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

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

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

   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 swap avec ✓ `SECOND (Ready)` sur serveur node1 et ✓ `PRIM (Ready)` sur serveur node2

   Dans le journal du nouveau `PRIM`, message :

   "Virtual IP <virtip of mirror> set"

3. Sur le serveur node2, `ipconfig` ou `ifconfig` (ou `ip addr show`) 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` :

```
<rfs>
  <replicated dir "/replicated"
mode="read_only" />
                                (ou
"C:\replicated")
</rfs>
```

1. Sur le serveur ✓ `PRIM (Ready)`, aller sous `/replicated` et créer 1 fichier `file1.txt`
2. Sur le serveur ✓ `SECOND (Ready)`, aller sous `/replicated` 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)`.

1. Le fichier `file1.txt` a été répliqué sur le serveur ✓ `SECOND (Ready)` sous `/replicated`
2. Echec car le répertoire répliqué `/replicated` est en lecture seule sur le serveur ✓ `SECOND (Ready)`
3. Le fichier `file2.txt` n'est pas répliqué dans `/replicated` 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 réintégré, message
   "Updating directory tree from /replicated"

   Et à la fin de la réintégration de `/replicated`, 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 d'un module miroir sur le serveur ✓PRIM (Ready)

- sur Windows, vérifier que la procédure spéciale d'arrêt des modules au shutdown a été réalisé (voir section 10.4 page 165)
- effectuer un shutdown du serveur ✓PRIM (Ready)
- vérifier dans le journal du serveur ✓SECOND (Ready) le message

  "Reason of failover: remote stop"
- le serveur ✓SECOND (Ready) devient ✓ALONE (Ready) ; l'application dans le script start_prim du module est redémarrée 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 gris
- après reboot du serveur arrêté, vérifier que le shutdown de l'OS a réellement appelé avec un shutdown du module avec le message

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

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

  "End of stop"
- 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 d'un module miroir sur le serveur ✓PRIM (Ready)

userconfig.xml :
`<heart>`
`  <heartbeat name="default" />`
`  <heartbeat 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 section 1.3.6 page 18
	➪ dans le journal du nœud ✓SECOND (Ready), message pour tous les heartbeats définis dans `userconfig.xml`
	"Resource heartbeat.default set to down by heart"
	"Resource heartbeat.flow set to down by heart"
	"Remote state UNKNOWN grey"
	"Reason of failover: no heartbeat "
	
	➪ les messages apparaissent après 30 secondes suite au 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 redémarrée sur le serveur ALONE avec le message dans son log
	"Script start_prim"
	
	➪ sur timeout dans la console web, l'ancien serveur ✓PRIM (Ready) devient gris
	
	➪ 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 reboot, message dans le journal:
	"Previous halt unexpected"
	
	➪ 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 d'adresse IP sur l'IP virtuelle virtip

```
<vip>
 <interface_list>
  <interface check="on"
arpreroute="on">
   <real_interface>
    <virtual_addr addr="virtip"
       where="one_side_alias"/>
   </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 tous les réseaux avec heartbeat (réseau default et repli)
2. reconnecter les réseaux

- 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 grey"
  "Local state ALONE Ready"

- cas de split brain : les 2 serveurs sont ✓ALONE (Ready) et exécute l'application démarrée dans start_prim
- lorsque les réseaux de heartbeat 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

- 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 page 86.

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 7.3 page 115)

  "Action start called by web@<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 7.3 page 115)

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

- le nœud arrêté exécute le script stop_both qui arrête l'application sur le serveur avec le message dans le log

  "Script stop_both"

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

  "End of stop"

  "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 7.3 page 115)

  "Action restart called by web@<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 et redémarre 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_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>
  <virtual_interface
type="vmac_invisible" >
    <virtual_addr
addr="virtip" where="alias"/>
  </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 de tous les serveurs :

  "Vitual IP <virtip of farm> set"

- Sur les 2 serveurs, `ipconfig` ou `ifconfig` (ou `ip addr show`) retourne virtip en alias de 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 adresse_ip_node1; ping adresse_ip_node2; ping virtip; arp –a
  adresse_ip_node1 00-0c-29-0a-5c-fc
  adresse_ip_node2 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

4.3.4.2 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 arpreroute="on">
  <virtual_interface
type="vmac_directed" >
    <virtual_addr
addr="virtip" where="alias"/>
  </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 de tous les serveurs :

  "Vitual IP <virtip of farm> set"

- Sur les 2 serveurs, `ipconfig` ou `ifconfig` (ou `ip addr show`) retourne virtip en alias de 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 ip1.1; ping ip1.2; ping ip1.20; 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
  ```

4.3.5 Test load balancing TCP sur une adresse virtuelle

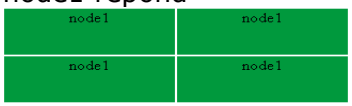
Le module ferme est dans l'état ✓`UP (Ready)` sur les 2 serveurs node1, node2.

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 la variable power dans `userconfig.xml`)

⇨ ✓`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 50%.

Exemple de logs avec node1 et node2

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

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

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

```
Bitmap 1:00000000:00000000:00000000:00000000:
ffffffff:ffffffff:ffffffff:ffffffff
Bitmap 2:ffffffff:ffffffff:ffffffff:ffffffff:
00000000:00000000:00000000:0000000
```

Les bits sont équitablement répartis entre les 2 serveurs

⇨ ✗`STOP (NotReady)` sur node2 : les sessions TCP sont servies par node1 lorsqu'on charge l'URL

Dans le journal de node1 :

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

256/256: tous les bits sont gérés par node1

`safekit -r vip_if_ctrl -l` sur node1:

```
Bitmap 1: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 les serveurs `ip1.1`, `ip1.2`.

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

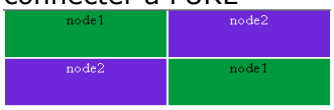
Sur la station externe:

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

2. Déconnecter le réseau entre `ip1.1` et `ip1.2`. Suivant l'endroit où se trouve la station externe, node 1 ou node 2 répond

ou

3. reconnecter le réseau et se connecter à l'URL



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

- avant split brain, état ✓`UP (Ready)` ✓`UP (Ready)`. Dans les ressources pour node1 et node2 : `FarmProto 50%`

  Dans les logs de node1 et node2:

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

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

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

```
Bitmap 1:00000000:00000000:00000000:00000000:
ffffffff:ffffffff:ffffffff:ffffffff
Bitmap 2:ffffffff:ffffffff:ffffffff:ffffffff:
00000000:00000000:00000000:0000000
```

  Les bits sont équitablement répartis entre les 2 serveurs

- après isolation réseau, split brain :

  Dans les ressources, pour node1 et node2 : `FarmProto 100%`

  dans le journal de node1 :

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

  256/256 : tous les bits sont gérés par node 1

  `safekit -r vip_if_ctrl -l` on node1:

```
Bitmap 1:ffffffff:ffffffff:ffffffff:ffffffff:
ffffffff:ffffffff:ffffffff:ffffffff
```

  dans le journal de node 2:

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

  256/256: tous les bits sont gérés par node 2

```
Bitmap 2:ffffffff:ffffffff:ffffffff:ffffffff:
         ffffffff:ffffffff:ffffffff:ffffffff
```

- après split brain lorsque le réseau est reconnecté entre `ip1.1` et `ip1.2`, 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.

En vmac_directed, les messages dans le journal et le résultat de `vip_if_ctrl` sont différents.

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

4.3.7.1 Prérequis réseau

Une adresse MAC Ethernet unicast 5a-fe-xx-xx-xx-xx est associée à l'adresse virtuelle ip1.20 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 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 page 74

4.3.7.2 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 ip1.20` : capture tous les paquets réseau
- tous les serveurs sont ✓`UP (Ready)`
- le monitoring réseau est lancé dans chaque serveur en filtrant sur ip1.20
- une console externe envoie un seul ping vers l'adresse IP virtuelle avec `ping -n (ou -c) 1 ip1.20`
- résultat : 1 paquet "ICMP: Echo: From ipconsole To ip1.20" envoyé et reçu par l'ensemble des serveurs
- résultat : il doit y avoir autant de paquets "ICMP: Echo Reply: To ipconsole From ip1.20" 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 ip1.20 icmp_seq=1
  64 bytes from ip1.20 icmp_seq=1 (DUP!)
  64 bytes from ip1.20 icmp_seq=1 (DUP!)
  64 bytes from ip1.20 icmp_seq=2
  64 bytes from ip1.20 icmp_seq=2 (DUP!)
  64 bytes from ip1.20 icmp_seq=2 (DUP!)
  ...
  ```

  Ce test peut être réalisé pendant plusieurs heures 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 ip1.20 >> /tmp/ping.txt`

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

- sur Windows, vérifier que la procédure spéciale d'arrêt des modules au shutdown a été réalisée : voir section 10.4 page 165
- effectuer un shutdown d'un serveur ✓UP (Ready)
- les autres serveurs restent ✓UP (Ready) et continuent à exécuter l'application
- sur timeout de la console web, l'ancien serveur ✓UP (Ready) devient gris
- 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"
- et vérifier que le module a été complètement arrêté avant le shutdown du serveur avec le dernier message

  "End of stop"
- 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 module ferme sur un serveur ✓UP (Ready)

- les autres serveurs restent ✓UP (Ready) et continuent à exécuter l'applicatif
- sur timeout dans la console web, l'ancien serveur ✓UP (Ready) devient gris
- 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 reboot, message dans le log

  "Previous halt unexpected"
- 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 section 4.4 page 86.

4.4 Tests des checkers communs à un miroir et une ferme

4.4.1 Test <errd>: checker de processus avec action restart ou stopstart

Dans userconfig.xml :

```
<errd>
<proc name="appli.exe" atleast="1"

action="restart "

class="prim "/>
</errd>
```

- name="appli.exe" atleast="1": au moins un processus "appli.exe" doit s'exécuter
- class="prim" (cas d'un module miroir) checker exécuté sur le serveur dans l'état ✓ (Ready) (i.e. PRIM ou ALONE) ; démarré après start_prim (arrêté avant stop_prim)
- class="both" (cas d'un module ferme) checker exécuté sur tous les serveurs dans l'état ✓ UP (Ready), démarré après start_both (arrêté avant stop_both)
- action="restart" : si "appli.exe" n'est pas présent, action restart qui exécute seulement les scripts stop_xx ; start_xx
- action="stopstart" : si "appli.exe" n'est pas présent, action stopstart qui arrête totalement le module puis le redémarre

Kill du 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), 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 log

  "Action start called automatically"

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

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

- avec les valeurs par défaut de maxloop="3" et loop_interval="24" (userconfig.xml <service>)
- au bout de 4 kills sur un même serveur, le module devient ✗ STOP (NotReady)
- message dans le journal avant l'arrêt :

  "Stopping loop"

4.4.2 Test <tcp> checker de l'applicatif local avec action restart ou stopstart

Dans `userconfig.xml` :

```
<tcp ident="id" when="prim ">
  <to addr="virtip" port="idport"
interval="10"

timeout="5" />
</tcp>
<failover>
<![CDATA[
tcpid_failure: if (tcp.id == down)
then stopstart();
]]>
</failover>
```

- le checker vérifie que l'application tcp lancée sur le port idport répond à des demandes de connexion
- `addr="ip.virtual"`, `port="idport"` : connexions TCP testées sur l'adresse IP virtip et sur le port TCP idport
- `interval="10"`, `timeout="5"` par défaut : test fait toutes les 10 secondes et avec un timeout de 5 secondes
- `class="prim"` (cas d'un module miroir) checker exécuté sur le serveur dans l'état ✓ `(Ready)` (i.e. `PRIM` ou `ALONE`) ; démarré après `start_prim` (arrêté avant `stop_prim`)
- `class="both"` (cas d'un module ferme) checker exécuté sur tous les serveurs dans l'état ✓ `UP (Ready)`, démarré après `start_both` (arrêté avant `stop_both`)
- action `restart()` : règle de failover par défaut ; si la connexion TCP locale échoue, action restart qui exécute seulement les scripts `stop_xx` ; start_xx
- action `stopstart()` : si la connexion TCP locale échoue, action stopstart qui arrête totalement le module puis le redémarre

Arrêter l'application qui écoute sur le port idport 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 tcp.id set to down by tcpcheck"
  "Action restart|stopstart from failover rule tcpid_failure "
- le module passe dans un état ↻ `(Transient)`, respectivement `PRIM`, `ALONE` ou `UP`
- 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

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

- avec les valeurs par défaut de `maxloop="3"` et `loop_interval="24"` (`userconfig.xml` <service>)
- au bout de 4 arrêts de l'application sur un même serveur, le module devient ✗ `STOP (NotReady)`
- message dans le journal avant l'arrêt :

  "Stopping loop"

4.4.3 Test <tcp> checker d'un service externe avec action wait

Dans `userconfig.xml` :

```
<tcp ident="id" when="pre">
  <to addr="ip.externe" port="idport"
                    interval="10"
timeout="5" />
</tcp>
<failover>
<![CDATA[
tcpid_failure: if (tcp.id== down) then
wait();
]]>
</failover>
```

- le checker vérifie que le service TCP externe (`ip.externe`, `idport`) répond à des demandes de connexion
- `interval="10"`, `timeout="5"` par défaut : test fait toutes les 10 secondes et avec un timeout de 5 secondes
- `when="pre"` checker lancé sur tous les serveurs après le script `prestart` (et arrêté avant `poststop`)
- si la connexion TCP échoue, le checker met la ressource `tcp.id` à `down`. La règle de failover sur le TCP checker exécute l'action wait qui arrête l'applicatif et met le module dans l'état `WAIT` en attente de `tcp.id` repositionné à `up` par le checker

Arrêter le service TCP (ip.externe, idport) sur le 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 tcp.id set to down by tcpcheck"
  "Action wait from failover rule tcpid_failure"
- dans tous les cas, le module devient ○ `WAIT (NotReady)` sur le serveur

Redémarrer le service TCP externe :

- messages dans le log

  "Resource tcp.id set to up by tcpcheck"
  "Transition `WAKEUP` from failover rule Implicit_`WAKEUP`"
- le module redevient ✓ `(Ready)`, respectivement dans l'état `SECOND`, `ALONE`, `SECOND` ou `UP`

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

Reproduire le test sur le même serveur

- avec les valeurs par défaut de `maxloop="3"` et `loop_interval="24"` (`userconfig.xml` <service>)
- au bout de 4 redémarrages sur un même serveur, le module devient ✗ `STOP (NotReady)`
- message dans le journal avant l'arrêt :

  "Stopping loop"

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"> sur une interface réseau locale avec action wait

Dans `userconfig.xml` :

```
<vip>
 <interface_list>
  <interface check="on">
      <!--
         définition d'une adresse IP
virtuelle
         sur le réseau default
       -->
  </interface>
 </interface_list>
</vip>
```

Règle de failover par défaut = `wait`

- Un checker vérifie que le câble Ethernet est connecté dans l'interface du réseau `ip.0` où est définie l'adresse IP virtuelle
- Si le câble est déconnecté, le checker met la ressource `intf.ip.0` à `down`. La règle de failover sur les interfaces checkers exécute l'action stopwait qui arrête l'applicatif et met le module dans l'état `WAIT` en attente de `intf.ip.0` repositionne 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

Retirer le câble Ethernet de la carte du réseau ip.0 sur le 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 intf.ip.default set to down by intfcheck"

  "Action wait from failover rule interface_failure"

  "Transition `WAIT`_TR from failover rule interface_failure"

- dans tous les cas, le module devient ○ `WAIT (NotReady)` sur le serveur

Remettre le câble :

- messages dans le journal

  "Resource intf.ip.0 set to up by intfcheck"
  "Transition `WAKEUP` from failover rule Implicit_`WAKEUP`"

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

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

Reproduire le test sur le même serveur

- avec les valeurs par défaut de `maxloop="3"` et `loop_interval="24"` (`userconfig.xml` <service>)
- au bout de 4 redémarrages sur un même serveur, le module devient ✗ `STOP (NotReady)`
- message dans le journal avant l'arrêt :

  "Stopping loop"

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

4.4.5 Test <ping> checker avec action wait

Dans `userconfig.xml`:

```
<ping ident="id" when="pre">
  <to addr="ip.device" interval="10"

timeout="5"/>
</ping>
```

Règle de failover par défaut = wait

- le checker vérifie qu'un composant externe (ex. : un routeur) avec l'adresse `ip.device` répond au ping
- `interval="10"`, `timeout="5"` par défaut : test fait toutes les 10 secondes et avec un timeout de 5 secondes
- `when="pre"` checker lancé sur tous les serveurs après le script `prestart` (et arrêté avant `poststop`)
- si le ping ne répond pas, le checker met la ressource `ping.id` à `down`. La règle de failover sur les pings checkers exécute l'action stopwait qui arrête l'applicatif et met le module dans l'état `WAIT` en attente de `ping.id` repositionné à `up` par le checker

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 ping_failure"
- dans tous les cas, le module devient ○ `WAIT (NotReady)` sur le serveur

Rétablir la liaison réseau :

- messages dans le journal

  "Resource ping.id set to up by pingcheck"
  "Transition `WAKEUP` from failover rule Implicit_`WAKEUP`"
- le module redevient ✓ `(Ready)`, respectivement dans l'état `SECOND`, `ALONE`, `SECOND` ou `UP`.

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

Reproduire le test sur le même serveur

- avec les valeurs par défaut de maxloop="3" et loop_interval="24" (`userconfig.xml` <service>)
- au bout de 4 redémarrages sur un même serveur, le module devient ✗ `STOP (NotReady)`
- message dans le journal avant l'arrêt :

  "Stopping loop"

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

Dans `userconfig.xml` du module `X`, test d'un autre module `othermodule`:

`userconfig.xml` du module `X`:

```
<module name="othermodule">
  <to addr="ip" interval="10"

timeout="5"/>
</module>
```

- le checker vérifie le module `othermodule` sur son adresse IP virtuelle ip
- `interval="10"`, `timeout="5"` par défaut : test fait toutes les 10 secondes et avec un timeout de 5 secondes

Si le module `othermodule` n'est pas démarré, le module `X` reste dans l'état `WAIT` en attente de son démarrage

Le module X réalise un stopstart lorsque le module othermodule est redémarré

Note : si le module X 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 X bloqué dans un état `WAIT` si l'action stopstart arrive pendant la transition `SECOND` vers `ALONE`

Arrêter le module othermodule. Et démarrer le module X sur tous ses serveurs :

- messages dans le journal du module X

  "Resource module.othermodule_ip set to down by modulecheck
  "Action wait from failover rule module_failure"

- le module X devient ○ `WAIT` `(NotReady)` sur tous les serveurs

Démarrer le module `othermodule` :

- messages dans le journal du module X

  "Resource module.othermodule_ip set to up by modulecheck"
  "Transition `WAKEUP` from failover rule Implicit_`WAKEUP`"

- le module X démarre sur tous ses serveurs en ✓ `(Ready)`

Faire `safekit restart -m othermodule`

- message dans le journal du module X : "stopstart called by modulecheck"
- le module X s'arrête puis redémarre

Reproduire le test sur le même serveur

- avec les valeurs par défaut de maxloop="3" et loop_interval="24" (`userconfig.xml` <service>)
- au bout de 4 redémarrages sur un même serveur, le module devient ✗ `STOP (NotReady)`
- dans le log, message avant l'arrêt :

  "Stopping loop"

4.4.7 Test <custom> checker avec action wait

Dans `userconfig.xml` :

```
<custom ident="id" when="pre"

exec="customscript" >
</custom>
```

- le script `SAFE/module/AM/bin/customscript` est un custom checker : une boucle avec un test sur une ressource
- `when="pre"` : custom checker lancé sur tous les serveurs après script `prestart` (arrêté avant `poststop`)

Gestion de la ressource `custom.id` pour réaliser l'action :

- Dans le script customscript :

  sur erreur : `SAFE/`safekit set -r custom.id –v down –i customscript

  sur succès : `SAFE/`safekit set -r custom.id –v up –i customscript

  Dans userconfig.xml :

```
<failover>
<![CDATA[
customid_failure: if (custom.id ==
down) then wait();
]]>
</failover>
```

- si le custom checker met la ressource à `down`, action wait qui arrête totalement le module puis le redémarre en mode ○ `WAIT (NotReady)` et en attente du passage à `up` de la ressource par le custom checker

Provoquer l'erreur testée par le custom checker sur 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 custom.id set to down by customscript"
  "Action wait from failover rule customid_failure"
  "Transition `WAIT`_TR from failover rule customid_failure"

- dans tous les cas, le module devient ○ `WAIT (NotReady)` sur le serveur

Réparer l'erreur testée par le custom checker :

- messages dans le journal :

  "Resource custom.id set to up by customscript"
  "Transition `WAKEUP` from failover rule Implicit_`WAKEUP`"

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

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

Reproduire le test sur le même serveur :

- avec les valeurs par défaut de `maxloop="3"` et `loop_interval="24"` (`userconfig.xml` <service>)
- au bout de 4 redémarrages sur un même serveur, le module devient ✗ `STOP (NotReady)`
- message dans le journal avant l'arrêt :

  "Stopping loop"

4.4.8 Test <custom> checker avec action restart ou stopstart

4.4.8.1 Action via une règle de failover

Dans `userconfig.xml` :

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

- le script `SAFE/module/AM/bin/customscript` est un custom checker : une boucle avec un test sur l'application intégrée dans les scripts
- `class="prim"` (cas d'un module miroir) checker exécuté sur le serveur dans l'état ✓ `(Ready)` (i.e. `PRIM` ou `ALONE`) ; démarré après `start_prim` (arrêté avant `stop_prim`)
- `class="both"` (cas d'un module ferme) checker exécuté sur tous les serveurs dans l'état ✓ `UP (Ready)` ; démarré après `start_both` (arrêté avant `stop_both`)

Gestion de la ressource custom.id pour réaliser l'action :

- Dans le script `customscript` :

  sur erreur : SAFE/safekit set -r custom.id –v down –i customscript

  sur succès : SAFE/safekit set -r custom.id –v up –i customscript

- Dans `userconfig.xml` :

```
<failover>
<![CDATA[
customid_failure: if (custom.id ==
down) then restart();
]]>
</failover>
ou
<failover>
<![CDATA[
customid_failure: if (custom.id ==
down) then stopstart();
]]>
</failover>
```

Provoquer l'erreur testée par le custom checker 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 custom.id set to down by customscript"

  "Action restart|stopstart from failover rule customid_failure"

  "Transition RESTART|STOPSTART from failover rule customid_failure"

- le module passe dans un état ↻ `(Transient)`, respectivement `PRIM`, `ALONE` ou `UP`.

- 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

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

- avec les valeurs par défaut de `maxloop="3"` et `loop_interval="24"` (`userconfig.xml` <service>)
- au bout de 4 redémarrages sur un même serveur, le module devient ✗ `STOP (NotReady)`
- dans le log, message avant l'arrêt :

  "Stopping loop"

4.4.8.2 Action directe dans le script

Dans `userconfig.xml` :

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

- le script `SAFE/module/AM/bin/customscript` est un custom checker : une boucle avec un test sur l'application intégrée dans les scripts
- `class="prim"` (cas d'un module miroir) checker exécuté sur le serveur dans l'état ✓ `(Ready)` (i.e. `PRIM` ou `ALONE`) ; démarré après `start_prim` (arrêté avant `stop_prim`)
- `class="both"` (cas d'un module ferme) checker exécuté sur tous les serveurs dans l'état ✓ `UP (Ready)`, démarré après `start_both` (arrêté avant `stop_both`)

Sur erreur, exécute restart ou stopstart

- dans le script `customscript` :

  sur erreur :

  SAFE/safekit restart -i customscript

  ou

  SAFE/safekit stopstart -i customscript

- action `restart` : exécute seulement les scripts `stop_xx` ; start_xx
- action `stopstart` : arrête totalement le module puis le redémarre

Provoquer l'erreur testée par le custom checker 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 :

- message dans le journal :

  "Action restart|stopstart called by customscript"

- le module passe dans un état ↻ `(Transient)`, respectivement `PRIM`, `ALONE` ou `UP`.
- 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

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

- avec les valeurs par défaut de `maxloop="3"` et `loop_interval="24"` (`userconfig.xml` <service>)
- au bout de 4 redémarrages sur un même serveur, le module devient ✗ `STOP (NotReady)`
- message dans le journal avant l'arrêt :

  "Stopping loop"

Note : sur une action directe dans le custom checker, le compteur loop 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

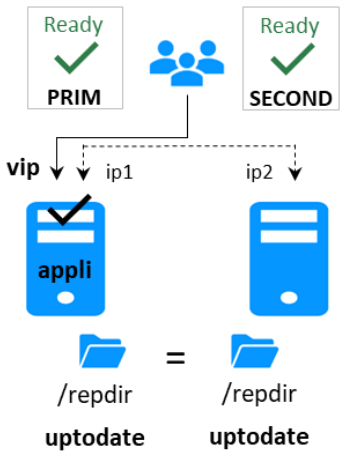
Pour tester un module miroir, voir section 4.2 page 72

Pour analyser un problème, voir section 7 page 111.

5.1 Mode de fonctionnement d'un module miroir

1. Fonctionnement normal

État stable : primaire avec secondaire



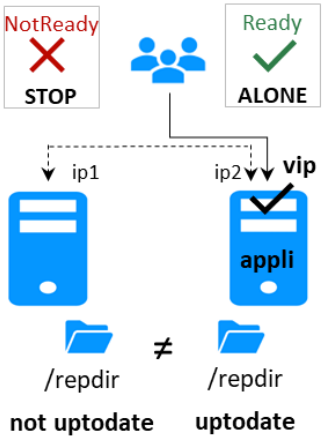
Sur la primaire :

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

La secondaire est prête à effectuer une reprise automatique pour devenir primaire.

2. Reprise automatique

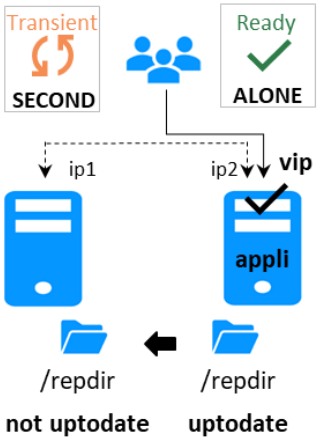
État stable : primaire sans secondaire



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

3. Réintégration après panne

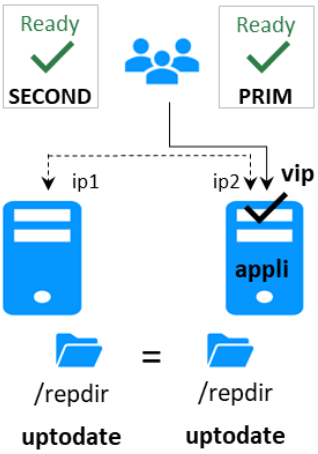
État transitoire : secondaire en cours de réintégration.



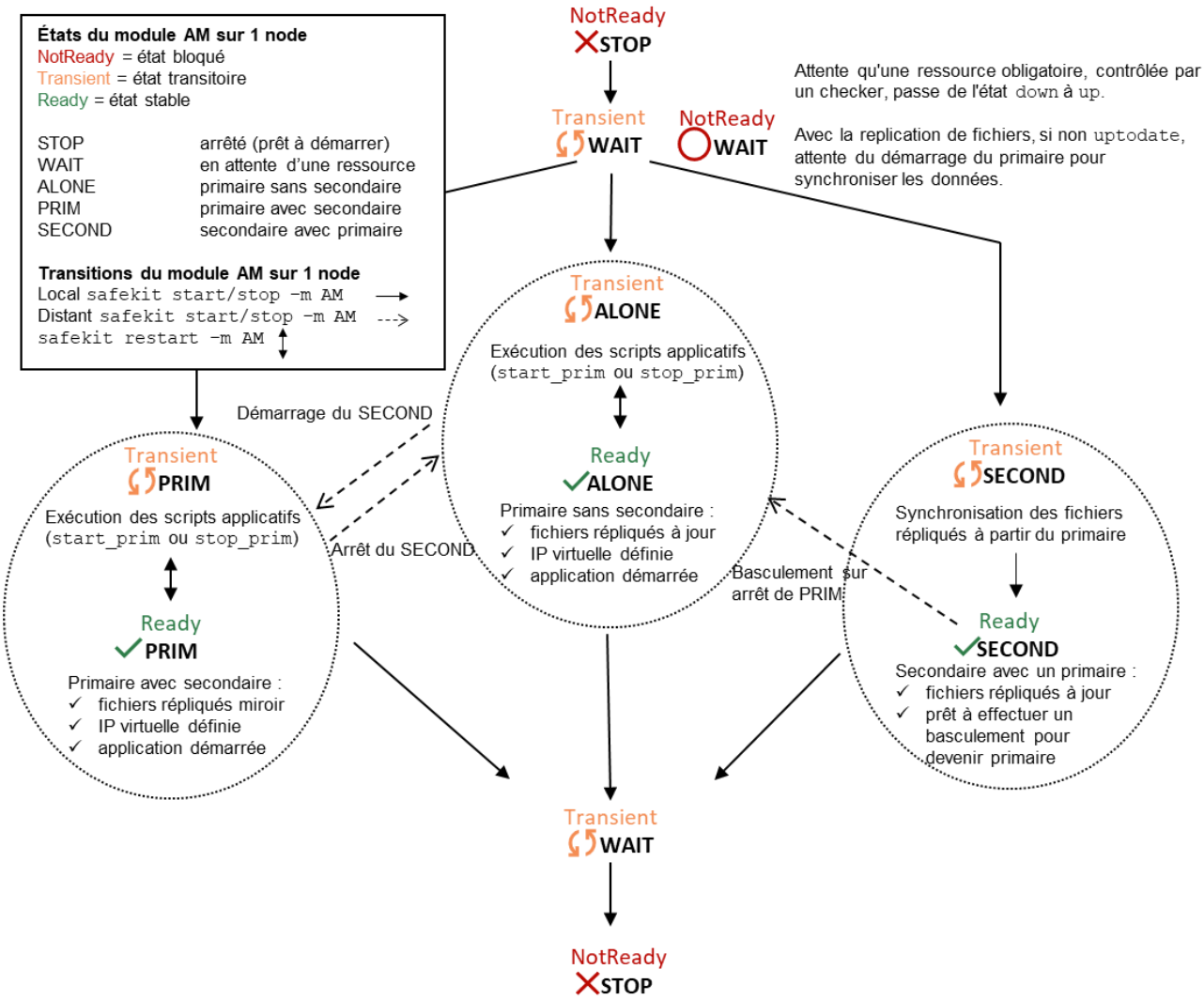
Synchronisation automatique des fichiers sans arrêt de l'application et mise à jour uniquement des fichiers modifiés sur le nœud primaire pendant que l'autre nœud était arrêté.

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`)

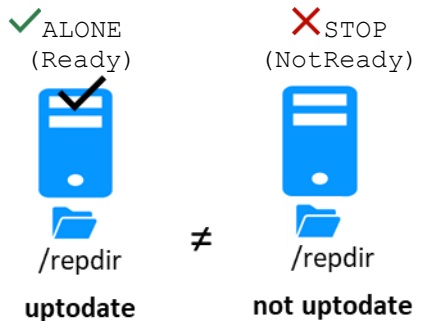
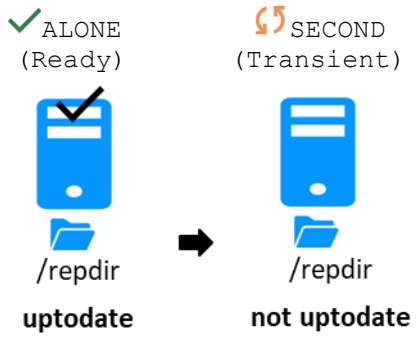
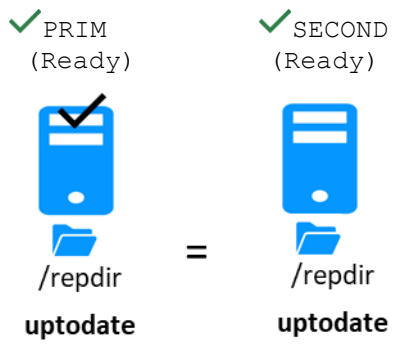
Au premier démarrage d'un module miroir, si les deux serveurs sont démarrés avec la commande `start`, ils se bloquent tous les deux dans l'état `WAIT (NotReady)` avec le message dans le journal : "Data may be not uptodate for replicated directories (wait for the start of the remote server)".
Au premier 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	STOP (NotReady) / STOP (NotReady) /repdir uptodate ≠ /repdir not uptodate
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 page 100 ➪ message dans le log: "Action prim called by web@<IP>/SYSTEM/root"	ALONE (Ready) / STOP (NotReady) /repdir uptodate ≠ /repdir not uptodate
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 web@<IP>/SYSTEM/root"	PRIM (Ready) / SECOND (Ready) /repdir uptodate = /repdir uptodate

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	

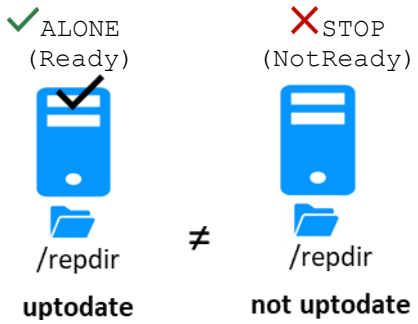
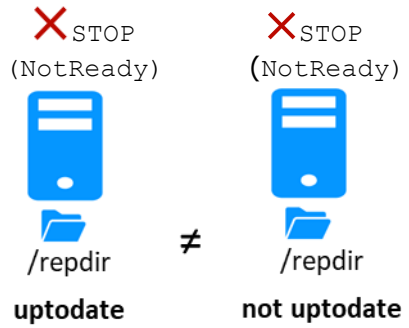
Le système de réplication maintient en plus sur chaque nœud la dernière date à laquelle les données étaient synchronisées. Cette date de synchronisation, nommée `synctimestamp`, est affectée à l'issue de la réintégration et évolue dans l'état ✓`PRIM (Ready)` et ✓`SECOND (Ready)`. Quand le module est 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`.

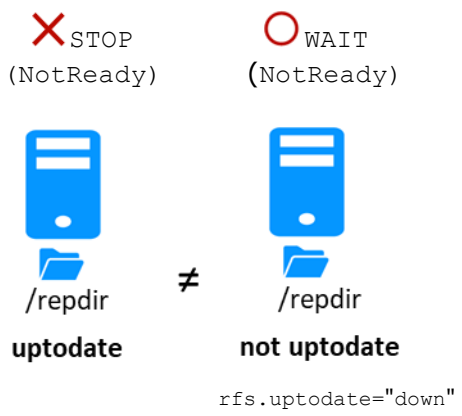
1. État initial ➪ node1 est primaire `ALONE` ➪ les répertoires sont à jour sur ce serveur ➪ le module est arrêté sur node2 ➪ node2 a des répertoires répliqués désynchronisés	
2. Commande `stop` sur node1 ➪ arrêt du serveur avec les répertoires à jour	

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



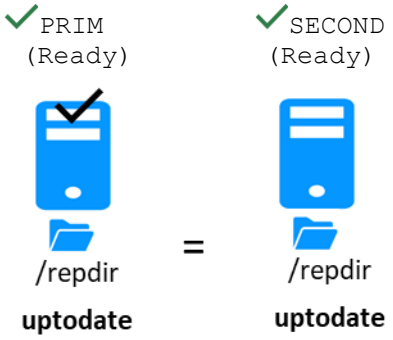
Voir aussi la section 5.9 page 105

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 web/ Contrôle sous le serveur `ALONE`. 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`" dans son log. `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"`	✓`ALONE (Ready)` ○`WAIT (NotReady)` /repdir ≠ /repdir **uptodate** **not uptodate** `rfs.degraded="up"` `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)`	✓`PRIM (Ready)` ✓`SECOND (Ready)` /repdir = /repdir **uptodate** **uptodate**

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

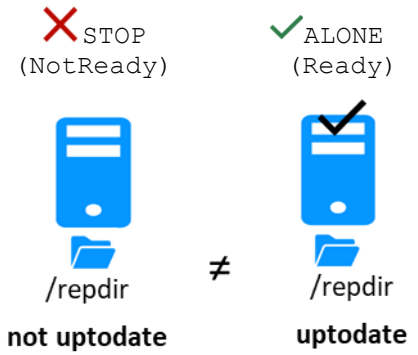
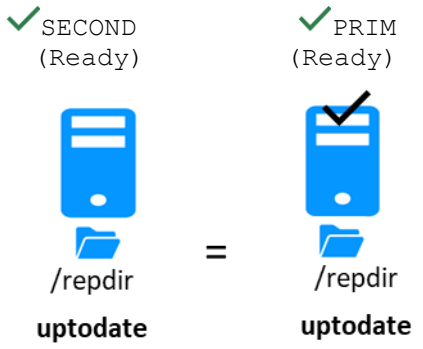
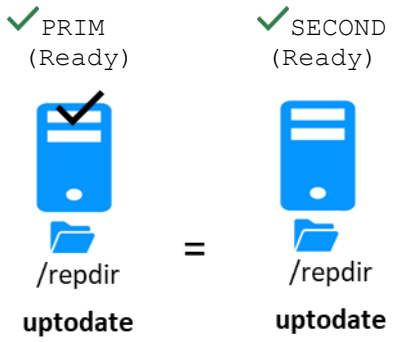
1. État initial ➪ le miroir est dans l'état stable node1 ✓ `PRIM (Ready)` – serveur 2 ✓ `SECOND (Ready)`	✓ `PRIM (Ready)` ✓ `SECOND (Ready)` /repdir = /repdir **uptodate** **uptodate**
2. Fonctionnement avec `failover="on"` ➪ si node1 anciennement `PRIM` rencontre une défaillance et s'arrête, node2 devient automatiquement primaire `ALONE` (mode par défaut)	✗ `STOP (NotReady)` ✓ `ALONE (Ready)` /repdir ≠ /repdir **not uptodate** **uptodate**
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 (Ref)` ➪ L'administrateur peut décider de forcer node2 à devenir primaire avec les commandes : `stop` ; `prim` sur node2	✗ `STOP (NotReady)` ○ `WAIT (NotReady)` /repdir = /repdir **not uptodate** **not uptodate**

Voir aussi la section 5.9 page 105

5.8 Serveur primaire par défaut (swap automatique après réintégration)

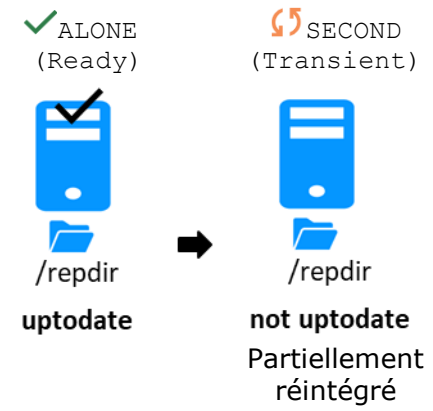
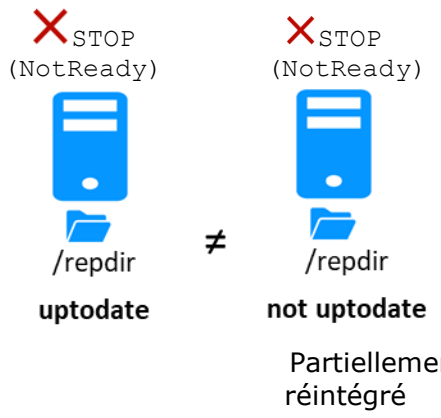
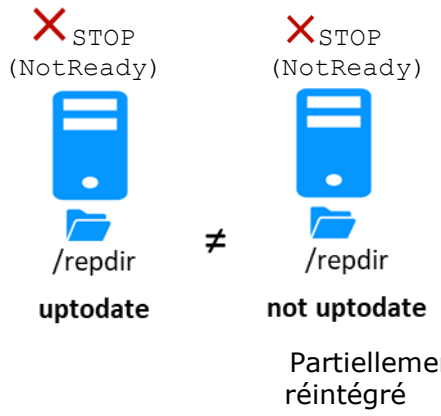
A la réintégration après panne, un serveur redevient par défaut secondaire. L'administrateur peut choisir de ramener l'application sur le serveur réintégré à un moment opportun avec la commande `swap`. C'est le comportement par défaut lorsque dans `userconfig.xml` <service> est défini sans la variable `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 -`	 la commande `prim` échoue car les données peuvent être corrompues

`m AM` (où AM est le nom du module) sur node2. Message dans le journal : "Action primforce called by SYSTEM/root"	

Note : La commande `primforce` force une réintégration complète des répertoires répliqués sur la secondaire lorsqu'elle est démarrée.

6. Administration d'un module ferme

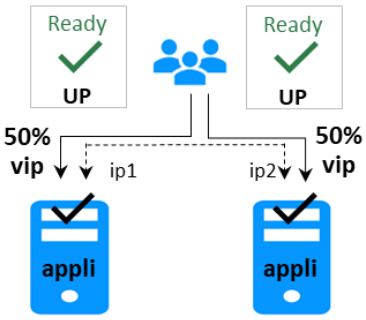
Pour tester un module ferme, voir section 4.3 page 79.

Pour analyser un problème, voir section 7 page 111.

6.1 Mode de fonctionnement d'un module ferme

1. Fonctionnement normal

État stable : 2 nœuds actifs.



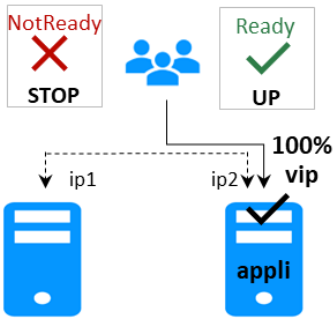
Sur tous les nœuds :

- ✓ 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.

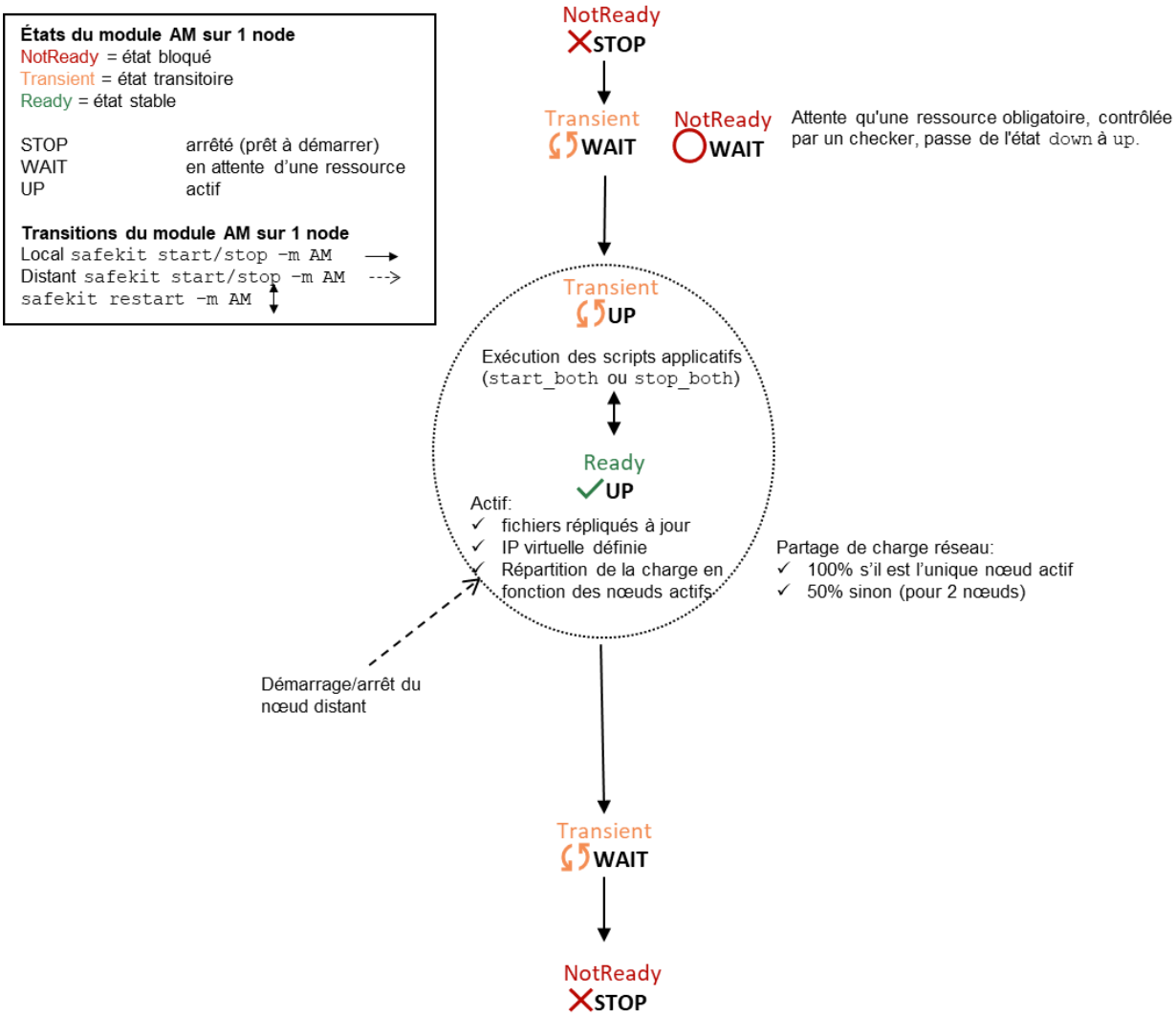


Sur arrêt du nœud distant, reprise automatique de toute la charge réseau.

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	STOP (NotReady) 0% STOP (NotReady) 0%
2. Commande `start` sur node1 et node2 ➪ message dans le journal des 2 serveurs : "farm membership: **node1 node2** (group FarmProto)" "farm load: **128/256** (group FarmProto)" "Local state `UP` Ready" ➪ ressource de chaque instance du module sur les 2 serveurs : `FarmProto 50%`	UP (Ready) 50% UP (Ready) 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 :

- ✓ que le navigateur et sa version sont bien supportés (dans certains environnements, Chrome fonctionne mieux qu'Internet Explorer)

- ✓ modifiez le paramétrage du proxy pour définir une connexion directe ou indirecte au serveur
- ✓ pour Internet Explorer, modifiez les paramètres de sécurité (ajoutez l'url dans les zones de sécurité)
- ✓ sur évolution de version de SafeKit, nettoyez le cache du navigateur comme décrit plus loin
- ✓ 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 :

- ✓ le pare-feu

  Si cela n'a pas encore été fait, exécutez la commande `SAFE/safekit 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 page 160.

- ✓ 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/bin/webservercfg -passwd pwd` pour initialiser (ou réinitialiser) cette configuration avec le mot de passe de l'utilisateur `admin`. Pour plus de détails, voir 11.2.1 page 179.

- ✓ la disponibilité du réseau et du serveur
- ✓ les services `safeadmin` et `safewebserver`

  Ils doivent être démarrés

- ✓ la configuration du cluster

  Exécutez la commande `safekit cluster confinfo` (voir section 9.3 page 146). 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 0 page 207)

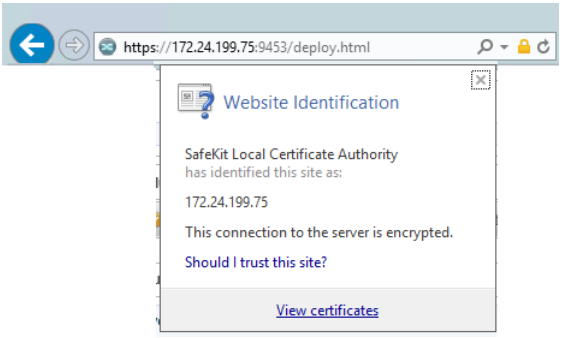
7.2 Problème de connexion HTTPS avec la console web

Si vous rencontrez des problèmes de connexion avec la console web en HTTPS, appliquez les contrôles et procédures ci-dessous :

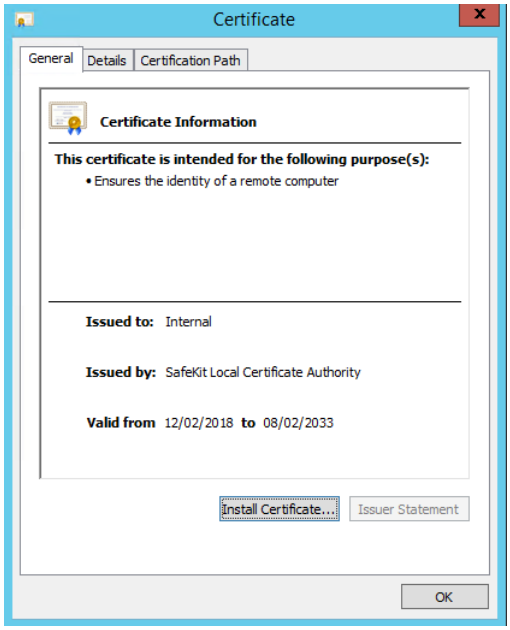
7.2.1 Contrôler les certificats serveurs

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



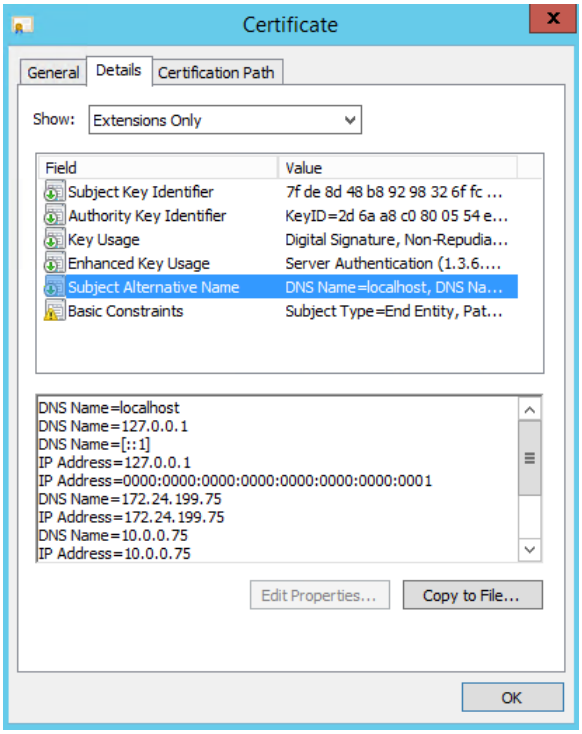
3. Vérifiez l'identité de l'émetteur qui doit être votre autorité de certification
4. Vérifiez la date de validité et la date de station de travail. Remettre la station à la bonne date si nécessaire
5. Vérifiez la date de validité. Si le certificat a expiré, vous devez le renouveler



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 0 page 207.

   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`

   La commande contrôle tous les certificats installés et échoue si une erreur est détectée
4. 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 :

- supprimer le fichier `SAFE/web/conf/ssl/httpd.webconsolessl.conf`
- exécuter `safekit webserver restart`

(`SAFE=C:\safekit` en Windows si System Drive=C: ; et `SAFE=/opt/safekit` en Linux):

Vous devrez également vider le cache du navigateur comme décrit en section 7.1.2 page 112.

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 alstable ✓ `(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 : voir l'index page 313.
- Les messages dans le journal après une commande administrateur sont :

"Action start called by web@<IP>/SYSTEM/root"
"Action stop called by web@<IP>/SYSTEM/root"
"Action restart called by web@<IP>/SYSTEM/root"
"Action swap called by web@<IP>/SYSTEM/root"
"Action stopstart called by web@<IP>/SYSTEM/root"
"Action forcestop called by web@<IP>/SYSTEM/root"

web@<ip>: via la console
SYSTEM: ligne de commande Windows
root: ligne de commande Linux

- Si "Stopping loop" apparaît dans le journal du module, voir section 7.11 page 120

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 section 10.9 page 173.

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 page 117
- si le module est ✕ WAIT, voir la section 7.9 page 118

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 page 117
- si le module est ○WAIT (NotReady), voir la section 7.9 page 118

7.8 Module ✕STOP (NotReady) : redémarrer le module

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

Vérifier que le module devient ✓ (Ready).

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 7.3 page 115).

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 stopwait.

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 page 95

"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 tcpid_failure"

- <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 page 264

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 7.3 page 115).

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 "Stopping loop" dans le journal du module : voir section 7.11 page 120

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, stopwait) 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` "Stopping loop"

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.

- 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.
- 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
- 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
- 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 page 115). 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 :

- Problème réseau

  Vérifier la configuration réseau

- Règles de pare-feu sur l'un ou les deux nœuds

  Voir section 10.3 page 160

- 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 page 205).

  - ✓ 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 9.3 page 146)

    Si la configuration du cluster SafeKit n'est pas identique, il faut réappliquer la configuration sur tous les nœuds comme cela est décrit en 3.2.2 page 43.

- 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 page 115). 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 10.5 page 166

- 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 10.5.3.1 page 167.

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 page 122 pour d'éventuelles solutions.

Voir aussi la section 4.3.6 page 83.

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 :

- 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>`
- 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)
- 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 :

- 4.3.4 page 80 pour le test de l'adresse virtuelle
- 4.3.5 page 82 pour le test du load-balancing
- 4.3.7 page 84 dans le cas d'une adresse MAC invisible

7.15 Problème après boot

Si vous rencontrez un problème après le boot, voir section 4.1 page 69.

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 13.2.3 page 213)

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 page 65. 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 config_2021_05_05_14_15_42 config_2021_07_08_16_34_05 config_2021_08_05_16_35_08	➪ `AM` Nom du module ➪ `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	➪ Journaux système En Linux, `last.txt` et `systemevt.txt` ou

systemevt.txt applicationevt.txt	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` :

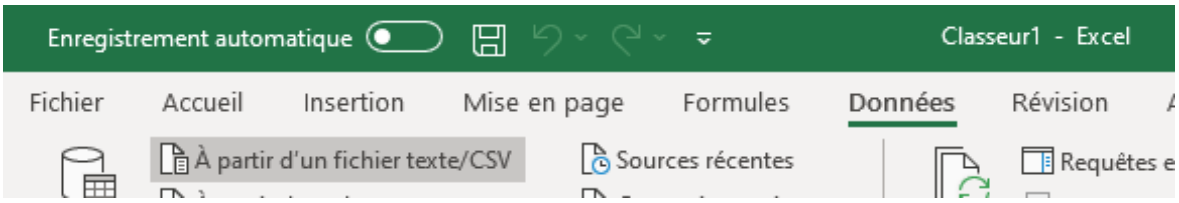
csv	
logverbose.csv resource.csv resourcelog.csv	➪ Journaux et états du module Journal verbeux Etat des ressources Historique de l'état des ressources
commandlog.csv	➪ Journaux et états du nœud Journal des commandes Liste des modules installés

modules.csv	Pour le support niveau 3
moduleslog.csv	Pour le support niveau 3
clusterstate.csv	

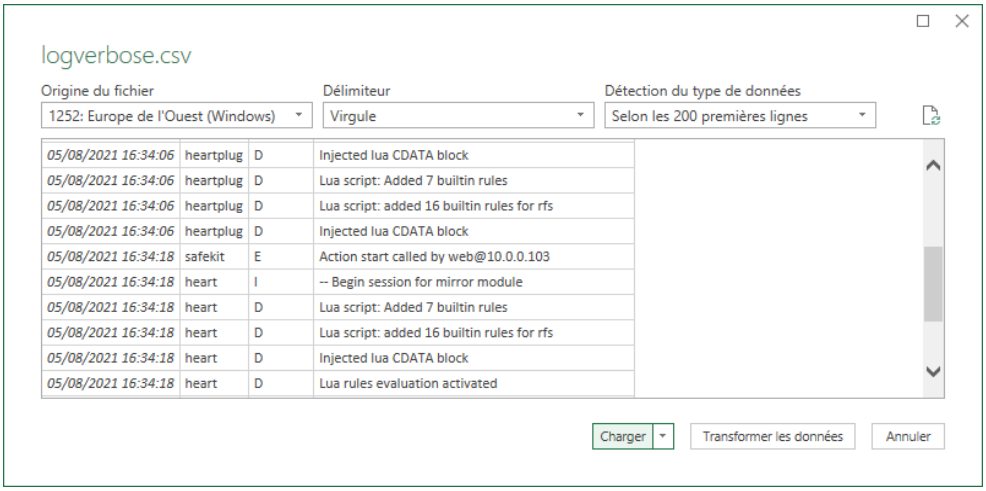
➪ Importer les fichiers csv dans Excel pour simplifier leur analyse

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).

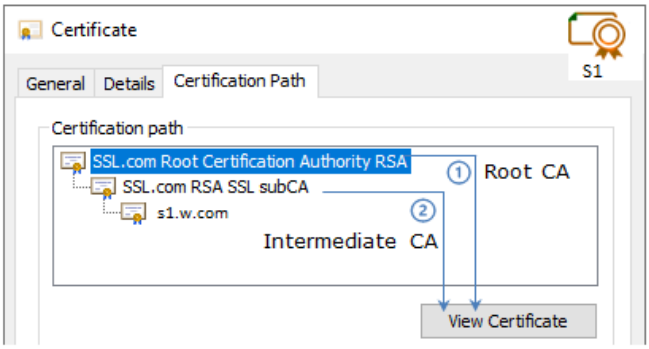
Si vous rencontrez des difficultés à récupérer ce fichier depuis la PKI, vous pouvez le construire en suivant les procédures décrites ci-dessous.

7.18.1 Exporter les certificats CA depuis des certificats publics

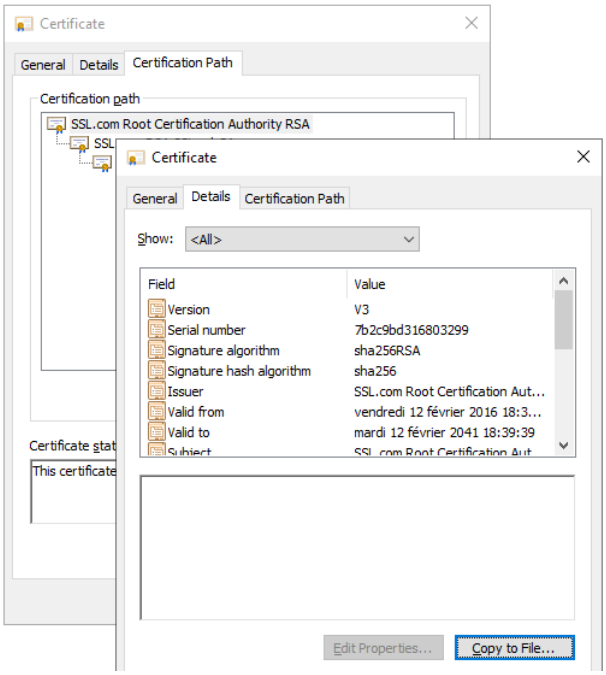
La procédure suivante explique comment construire, le fichier `combined.cer`, la chaîne de certificats pour les Autorités de Certification racine et intermédiaires d'un certificat public.

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 :

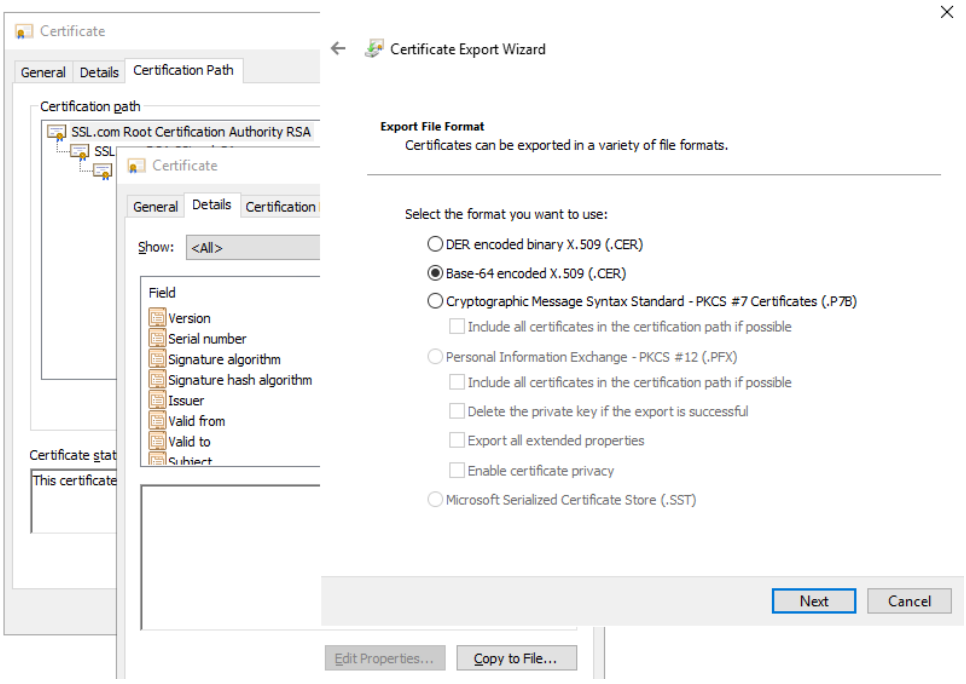
1. Copier le fichier .crt (ou .cer) sur une station de travail Windows
2. Double cliquer sur le fichier pour l'ouvrir avec « Extension noyau de chiffrement »
3. Cliquer sur l'onglet « Chemin d'accès de certification » pour afficher l'arbre des autorités de certification
4. Sélectionner une entrée (de haut en bas en excluant la feuille)



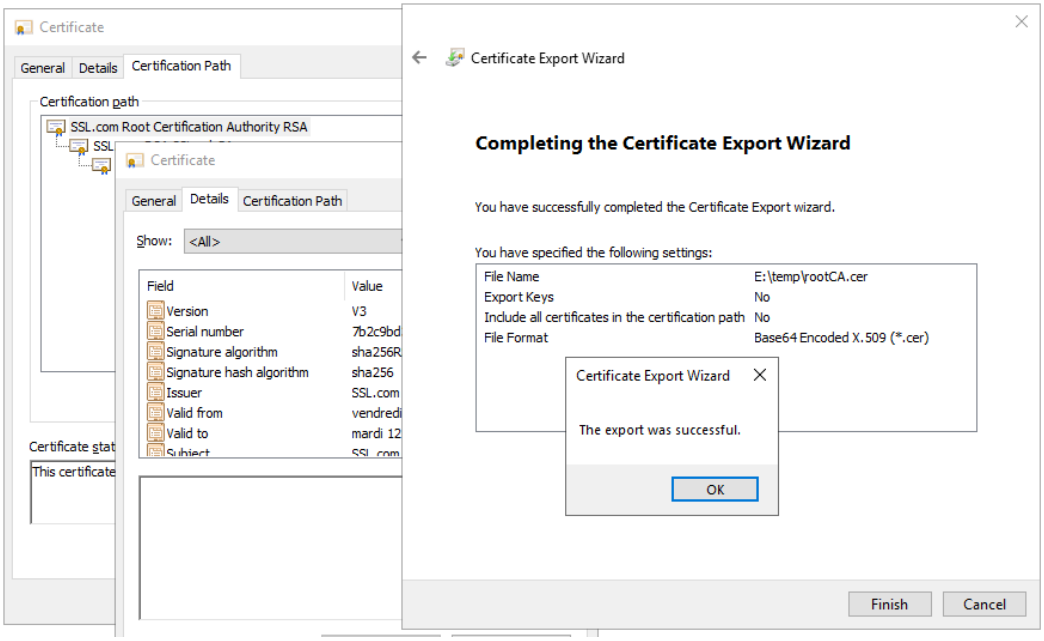
5. Cliquer sur « Afficher le certificat ». Une nouvelle fenêtre s'ouvre pour le certificat sélectionné
6. Dans cette nouvelle fenêtre, sélectionner l'onglet « Details » puis cliquer sur « Copier dans un fichier »



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
bRbjaT7JD6MBidAWRCJWC1R/5etTZwWwWrRCrzvIHC7WO6rCzwu69a+l7ofCKlWs
y702dmPTKEdEfwhgLx0LxJr/Aw==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIF3TCCA8WgAwIBAgIIeyyb0xaAMpkwDQYJKoZIhvcNAQELBQAwfDELMAkGA1UE
BhMCVVMxDjAMBgNVBAgMBVRleGFzMRAwDgYDVQQHDAdIb3VzdG9uMRgwFgYDVQQK
oYYitmUnDuy2n0Jg5GfCtdpBC8TTi2EbvPofkSvXRAdeuims2cXp7lNIWuuA8ShY
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 page 313
➪ voir section 8.5 page 136 qui décrit le Call Desk

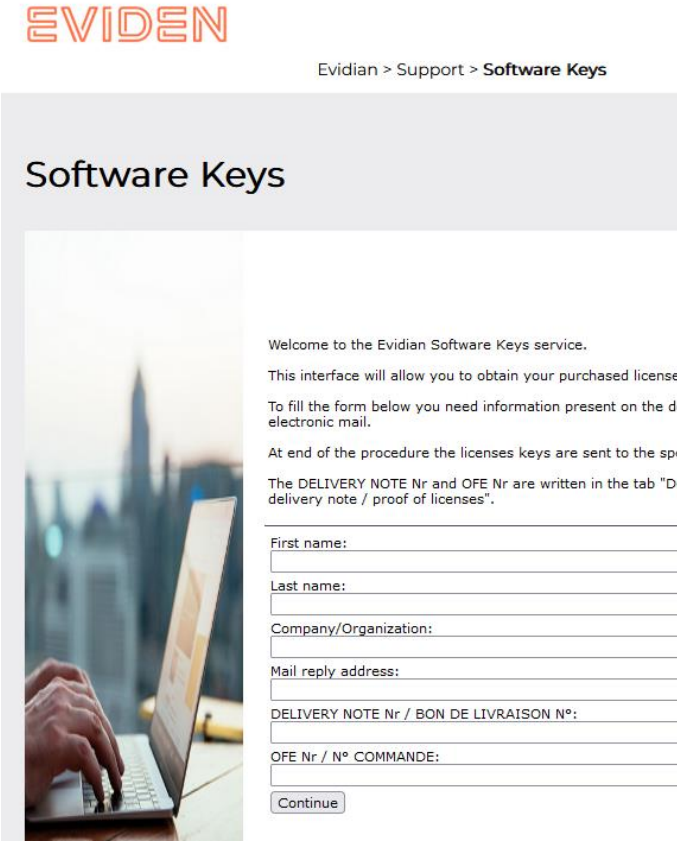
8. Accès au support Evidian

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

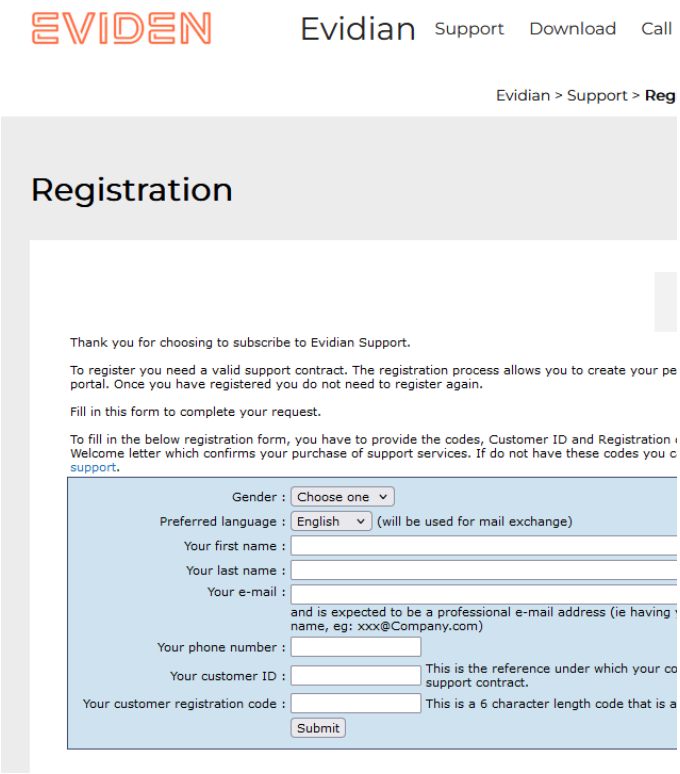
8.2 Clés de licence permanentes

- 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 section 2.1.5 page 29



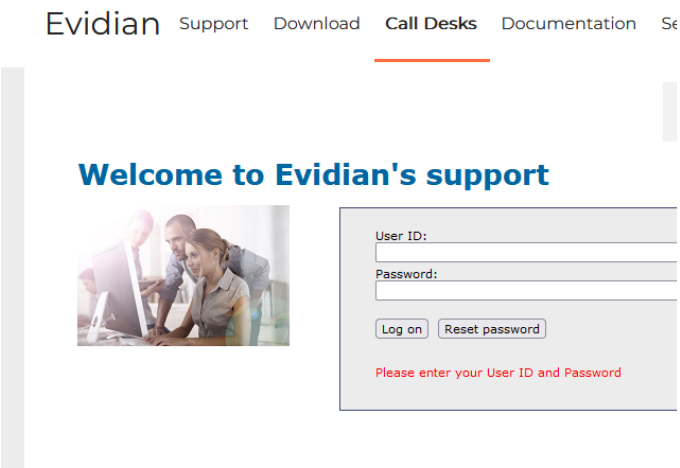
8.3 Créer un compte

- 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

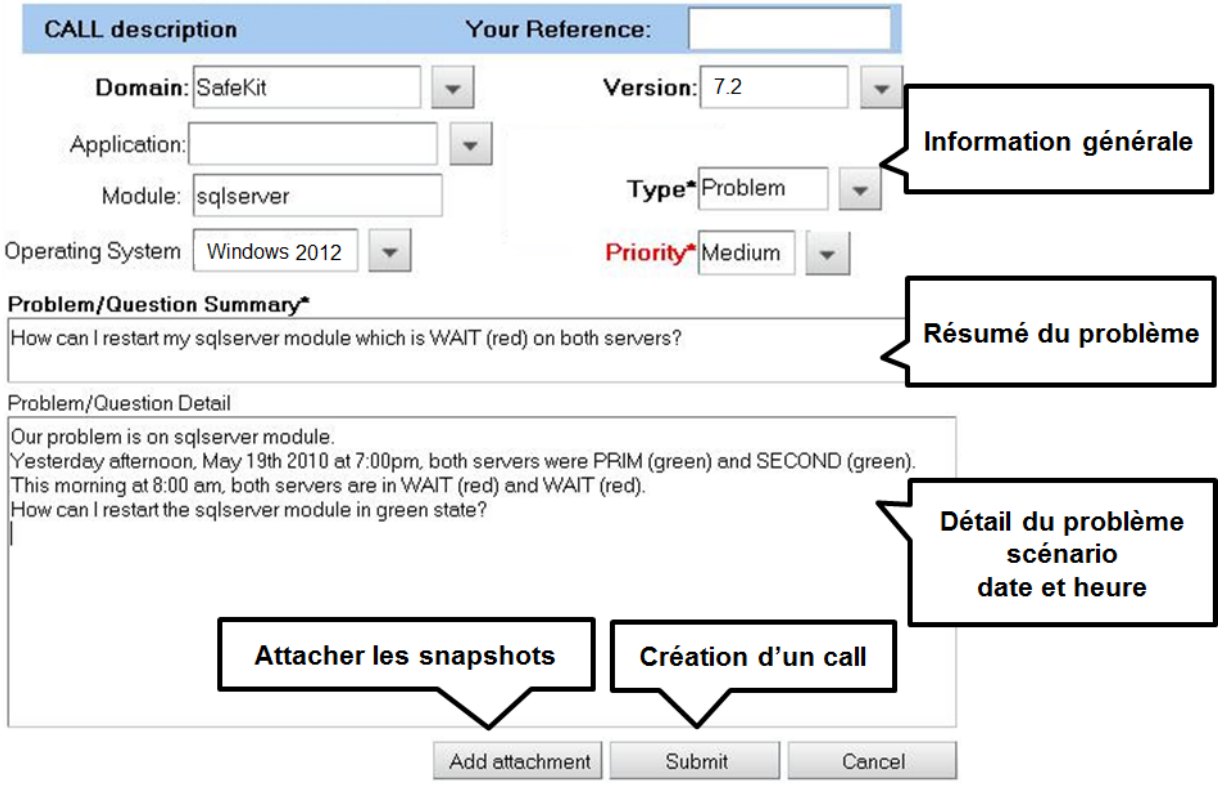
8.5.1 Les opérations du Call Desk

- https://www.evidian.com/support/call-desk/

  Call desk : outil pour remonter un problème au support avec 2 opérations principales

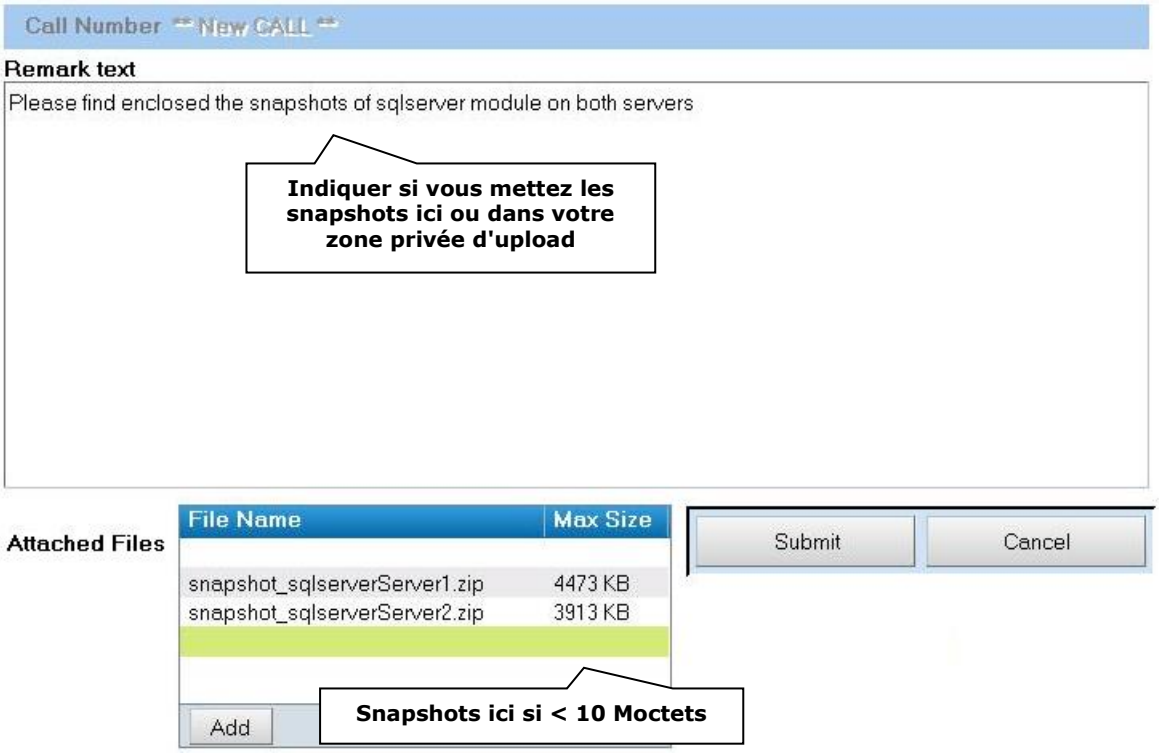
- Création d'un Call
- Recherche d'un Call et échange avec le support sur un Call

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 section 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 page 65
- 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 page 141

8.5.4 Consultation des réponses au Call et échange avec le support

Call Number: EVD000000034997 Created: 20/05/2010 10:21:38

Domain: SafeKit
Version: 7.2
Application:
Module: sqlserver
Operating System: Windows 2012

Status: Closure requeste
Type: Problem
Priority: Medium
Support responsible: Dominique Pires

Request for Closure | Add Remark | Close

Remark Text — Hide Remark text

To deconfigure the checker in the module, you must put this checker in commentary in the file userconfig.xml .
For that :
- edit the file userconfig.xml
- retreive the definition of the checker : it is defined like that :

```
<check>
  <ping
      ident="<checker name> "
    >
  <to
     addr="<IP address>"
    />
  </ping>
</check>
```

Ajouter un remarque pour poursuivre l'échange avec le support

Echange entre le support Evidian et le client jusqu'à la fermeture du call

Remark List

6 entries returned — Preferences — Refresh

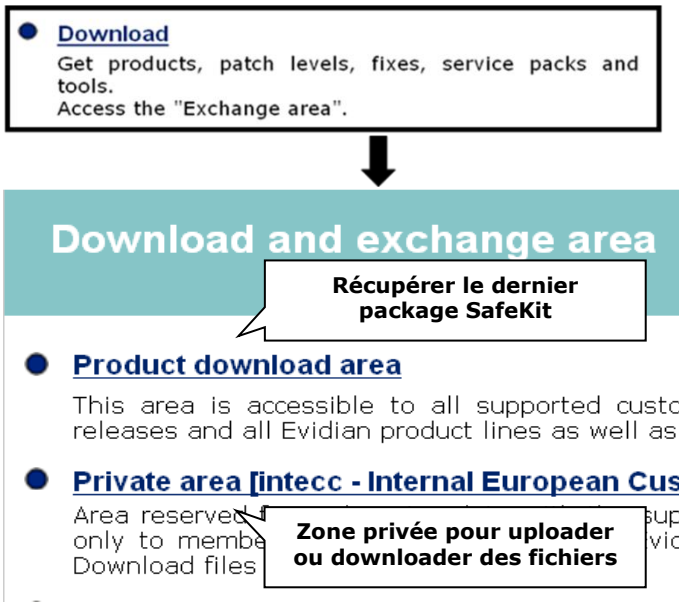
Date	Group	Submitter	Short Description
20/05/2010 15:07:54	CUST	rochat	Closure requested by rochat
20/05/2010 15:07:47	CUST	rochat	Thank you! The sqlserver module is restarted in PRIM (green) - SECOND (green)
20/05/2010 14:59:58	SUP	Dominique Pires	To deconfigure the checker in the module, you must put this checker in commentary in the file userco
20/05/2010 14:22:52	CUST	rochat	The pinged component has been removed last night. How can I deconfigure the checker in the module?
20/05/2010 13:56:13	SUP	Dominique Pires	According the logs, it seems that the 2 servers are in WAIT state, because the ping checkers defined
20/05/2010 10:19:08	CUST	rochat	Please find enclosed the snapshots of sqlserver module on both servers

- 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

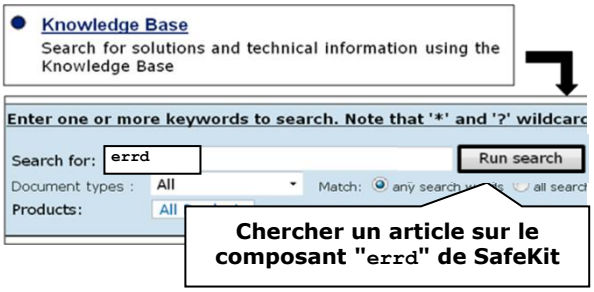
- Aller dans <Version 8.2>/Platforms/<Your platform>/Current versions
- Télécharger le package 64-bits SafeKit
- Pour plus d'information sur l'installation, la documentation et l'upgrade, voir section 2 page 25

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 page 65 pour la prise de snapshot
- Voir aussi 8.5.3 page 138 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>`
- En Linux
  4. Ouvrir une console Shell en tant que root
  5. Aller à la racine du répertoire d'installation de SafeKit `SAFE` (par défaut `SAFE=/opt/safekit`)

     `cd /opt/safekit`
  6. Exécuter `./safekit <arguments>`

9.1 Commandes distribuées

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 à un serveur.
L'interface ligne de commandes globale requiert l'exécution du service web Safekit sur chacun des serveurs de la liste (voir section 10.6 page 168).

safekit -H <url> [,<url,...] <action> <arg>	Exécute l'action sur les serveurs spécifiés par la liste d'URL
	
	Il est également possible de spécifier en guise d'url une liste de nom de serveurs (tels qu'ils apparaissent dans le fichier `cluster.xml`). Les URLS sont construites automatiquement en `https:9453` ou `http:9010` en fonction du contenu de `SAFE/web/conf/ssl`.
	
	La syntaxe –H "*" désigne tous les nœuds déclarés dans `cluster.xml`.
	
	Pour surcharger le protocole et port par défaut, spécifier comme premier élément un élément de la forme '[<protocol>:<port>]'. La partie ':<port>' est optionnelle. <protocol> peut être 'http' ou 'https'. Le port par défaut pour le protocole HTTP est 9010.
	
	Exemple : `safekit -H http://192.168.0.2:9010,http://192.168.0.3:9010 module list`
	
	`safekit –H "[http],*" module list`
	
	`safekit –H "*" module list`
	
	`safekit –H "[https:9500],server1,server2" module list`

safekit [-H <url>[,...]] -E <module>	Exporte le <module> localement installé sur les serveurs spécifiés par -H. Cette commande réalise les actions suivantes : ➪ crée <module>.safe à partir du module local SAFE/modules/<module> et note son id ➪ transfère et installe <module>.safe sur la liste de serveurs ➪ positionne l'id local du module sur les serveurs distants ➪ si le module était configuré localement, configure le module sur les serveurs distants Exemple : `safekit -E farm` exporte le module ferme local vers la liste des serveurs spécifiés dans `SAFEVAR/default_cluster.txt` (voir ci-dessus pour un exemple de `default_cluster.txt`)

safekit [-H <url>[,…] -G	Déploie les fichiers de configuration du cluster locaux sur tous les serveurs spécifiés par –H. Cette commande exécute les actions suivantes : ➪ Collecte le contenu du répertoire SAFEVAR/cluster ➪ Transfère les fichiers collectés dans le répertoire SAFEVAR/cluster du serveur cible. ➪ Déclenche le rechargement de la configuration de safeadmin.

9.2 Commandes de boot et shutdown

Utilisez les commandes suivantes pour démarrer/arrêter les services SafeKit, configurer le démarrage automatique au boot des services et des modules, arrêter tous les modules en cours d'exécution.

En Windows, vous aurez peut-être également besoin d'appliquer la procédure décrite en 10.4 page 165.

safeadmin (Windows)	Service principal de SafeKit obligatoire et démarré automatiquement au boot. safeadmin peut être contrôlé dans l'interface Services de Windows
service safeadmin start (Linux)	Service principal de SafeKit obligatoire et démarré automatiquement au boot
safekit boot –m AM [on | off | status]	Démarre automatiquement au boot ou non le module AM ("on" ou "off" ; par défaut "off") Sans l'option –m AM, safekit boot status liste l'état de tous les modules au boot Important: 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.
safekit webserver [start | stop | restart]	Contrôle le démarrage/arrêt/redémarrage du service safewebserver. Ce service est utile à la console SafeKit, aux checkers de <module> et aux commandes distribuées. La commande lance des processus httpd et attend le démarrage des processus.

`safekit boot [webon | weboff | webstatus]`	Contrôle le démarrage automatique au boot du service `safewebserver` ("on" ou "off" ; par défaut "on")
`safekit safeagent [start | stop | restart | check]`	En Windows : Contrôle le démarrage/arrêt du service `safeagent` qui met en œuvre un agent SNMP SafeKit.
`safekit boot [snmpon | snmpoff | snmpstatus]`	En Windows : Contrôle le démarrage automatique au boot du service `safeagent` ("on" ou "off" ; par défaut "off")
`safekit shutdown`	Stoppe tous les modules en cours d'exécution et attend leur arrêt complet

9.3 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. Exemple : `safekit cluster config /tmp/newcluster.xml` Note : 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. Si cette commande est appelée avec le paramètre `lock`, les futurs appels ne seront autorisés que si le paramètre `unlock` est utilisé.
`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 :
	`safekit cluster confinfo`
	`Nœud Signature Date Verrou`
	`rh6server7 6f1032b11a7b2 … 33e67c 2016-05-20T17:06:45 0`
	`rh7server7 6f1032b11a4e0 … 33e67c 2016-05-20T17:06:45 0`
	
	Note: 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 "[http],*" -G`	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).
`safekit -H <url>[,<url>] -G`	Distribue la configuration locale du cluster et les clés de chiffrement associées lorsqu'elles existent, sur les serveurs spécifiés dans la liste d'url. Exemple : `safekit -H http://192.168.1.1:9010,http://192.168.1.2:9010 -G`

9.4 Commandes de contrôle des modules

Les commandes s'appliquent au module nommé `AM`, passé en argument avec l'option `-m`.

`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 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 stopstart -m AM`	Contrairement à la commande `restart`, la commande `stopstart` provoque l'arrêt complet du module et son redémarrage. Si le module était `PRIM`, il y a basculement du module `PRIM` sur l'autre serveur Note: Equivalent à `safekit stop -m AM; safekit start -m AM`
`safekit forcestop -m AM`	Force l'arrêt du module lorsque des ressources sont gelées

`safekit restart -m AM`	Applique les scripts d'arrêt puis de démarrage de l'application : il n'y a pas de basculement sur l'autre serveur si le module est `PRIM`
`safekit swap [nosync] -m AM`	Module miroir uniquement Echange 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 page 98
`safekit errd suspend -m AM` `safekit errd resume -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.

`safekit checker off -m AM` `safekit checker on  -m AM`	Arrête ou démarre l'ensemble des checkers (interface, TCP, IP, custom, etc ...). Cette commande est utile lors d'opérations de maintenance ; lorsqu'il est connu que les checkers vont détecter une panne suite à 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. ✓ Un effet de bord est l'exécution de la commande `safekit update`
`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 page 213). 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. ✓ Un effet de bord de cette commande est l'exécution de la commande `safekit update`

9.5 Commandes de surveillance des modules

Les commandes s'appliquent au module nommé `AM`, passé en argument avec l'option `-m`.

`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 page 118 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 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.

9.6 Commandes de configuration des modules

safekit config -m AM	A exécuter après avoir modifié dans SAFE/modules/AM : userconfig.xml, start_prim/both ou stop_prim/both (miroir/ferme) Demande à chaque plugin défini dans userconfig.xml <errd>, <vip>, <rfs>, <user>... de prendre en compte la nouvelle configuration du module Cette commande peut être utilisée dans les états ALONE (Ready), ou STOP et WAIT (NotReady). Dans l'état STOP l'ensemble de la configuration peut être changée. Dans les états ALONE et WAIT, 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 page 211.
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 -H <url>[,<url>] -E AM	Distribue la configuration locale du module AM et les clés de chiffrement associées lorsqu'elles existent, sur les serveurs spécifiés dans la liste d'url. Ex: safekit -H http://192.168.1.1:9010,http://192.168.1.2:9010 -E mirror
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 nomdumodule.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

9.7 Commandes de support

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 7.16 page 124 Pour envoyer les snapshots au support, voir section 8 page 133
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. Important: 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.

9.8 Exemples

9.8.1 Configuration du cluster

Voir 12.2.2 page 208.

9.8.2 Configuration d'un nouveau 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. 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`

4. Éditer la configuration du module et les scripts sous `SAFE/modules/AM/conf` et `SAFE/modules/AM/bin`
5. Exécuter `safekit module genkey -m AM` ou `safekit module delkey -m AM`

   Pour créer ou détruire les clés d'encryption 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`).

9.8.3 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.zi`

   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. Note : 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 9.7 page 154

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 Processus et services SafeKit

Services SafeKit	Processus par module	
safeadmin (processus safeadmin): service principal et obligatoire	heart : gère les procédures de reprise	vipd : synchronise une ferme de serveurs
safewebserver (processus httpd) : service pour la console, les <module> checkers, les commandes distribuées	errd : gère la détection de la mort des processus	nfsadmin, nfsbox, reintegre : réplication et réintégration de fichiers en temps réel
safeagent (processus safeagent) : agent SNMP SafeKit (optionnel, uniquement en Windows)	checkers (ipcheck, intfcheck, …)	

Pour la liste détaillée des processus et ports de SafeKit, voir les sections 10.3.3.1 page 161 et 10.3.3.2 page 163.

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

10.3.1 Paramétrage du pare-feu en Linux

Si la configuration automatique du pare-feu a été choisie 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 choisie, vous devez configurer le pare-feu manuellement ou utiliser la commande safekit `firewallcfg`. Elle insère (ou supprime) les règles de pare-feu requises par les processus de base SafeKit (services `safeadmin` et `safewebserver`) et les processus des modules pour communiquer avec leurs homologues du cluster. Les administrateurs doivent s'assurer de l'absence de conflit avec une politique locale avant d'appliquer ces règles.

`safekit firewallcfg add` `safekit firewallcfg del`	Ajout (ou suppression) des règles pour le pare-feu firewalld ou iptable pour les ports des services `safeadmin` et `safewebserver` ➪ `SAFE/safekit firewallcfg add` ajout des règles pour les services `safeadmin` et `safewebserver` ➪ `SAFE/safekit firewallcfg del` suppression des règles pour les services `safeadmin` et `safewebserver`

`safekit firewallcfg add AM` `safekit firewallcfg del AM`	Ajout (ou suppression) des règles pour le pare-feu firewalld ou iptable pour les ports des modules SafeKit ➪ `SAFE/safekit firewallcfg add AM` ajout des règles pour le module nommé `AM` Important: Cette commande doit être exécutée après la première configuration du module, puis aux configurations suivantes si celles-ci modifient les ports utilisés (à vérifier avec la commande `safekit module getports -m AM`). ➪ `SAFE/safekit firewallcfg del AM` suppression des règles pour le module nommé `AM`

10.3.2 Paramétrage du pare-feu en Windows

En cas d'utilisation du pare-feu du système d'exploitation (pare-feu Microsoft), vous pouvez utiliser la commande safekit `firewallcfg`. Elle insère (ou supprime) les règles de pare-feu requises par les processus des services SafeKit (`safeadmin`, `safewebserver`, `safeacaserv et safeagent`) et les processus des modules pour communiquer avec leurs homologues du cluster. Les administrateurs doivent s'assurer de l'absence de conflit avec une politique locale avant d'appliquer ces règles.

`safekit firewallcfg add` `safekit firewallcfg del`	Ajout (ou suppression) des règles pour le pare-feu de Microsoft ➪ `SAFE/safekit firewallcfg add` ajout des règles pour les services SafeKit et les modules `SAFE/safekit 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 les commandes distribuées
  - ✓ service `safecaserv` (processus `httpd`)

    service web pour sécuriser la console web avec la PKI de SafeKit (optionnel)
  - ✓ En Windows : service `safeagent` (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 section 12.1.3 page 206.

- `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 section 10.6 page 168 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 section 11.3.1.9.4 page 191.

  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.

- `safeagent` (Windows uniquement, optionnel)
  Accès UDP, local et distant, sur le port 3600 par défaut. Pour la définition de la valeur du port, voir section 10.8 page 172.

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`.

La commande `safekit module getports -i ID` liste les ports qui peuvent être utilisés par le module ayant pour id `ID` (il n'est pas nécessaire que ce module soit installé, et si le module n'est pas configuré, la liste rendue sera un sur-ensemble des ports réellement utilisés par le module).

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 :
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

SAFE/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.

Important

Si vous utilisez l'agent SNMP de SafeKit, adaptez la procédure suivante pour positionner le démarrage manuel du service `safeagent` 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 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 10.5.3.1 page 167 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 0 page 167 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 9.6 page 152). 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 page 115. Cette ressource se nomme `usersetting.encryption`.

10.5.1 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 page 46).

10.5.2 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 9.6 page 152.

10.5.3 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).

10.5.3.1 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.

10.5.3.2 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
1. Exécuter `safekit module genkey -m AM`
3. Supprimer le fichier `SAFE/modules/AM/conf/modulekey.p12`
4. 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`
5. Aller dans le répertoire SAFE`/web/bin`
6. Générer votre certificat à l'aide de la PKI en spécifiant dans le sujet : `"/O=SafeKiModule/CN=mirror"`
7. Copier les fichiers générés `pkey.key` et `cert.crt` dans le répertoire `SAFE/web/bin`
8. 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
9. Supprimer les fichiers `pkey.key` et `cert.crt`
10. Déplacer le fichier `modulekey.p12` sous `SAFE/modules/AM/conf`
11. Aller dans le répertoire `SAFE`
12. 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 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 page 37)
- l'interface en ligne des commandes distribuées sur le cluster (voir section 9.1 page 143)
- les checkers de type <module> (voir section 13.16 page 263)

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 9.2 page 145.

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.

10.6.1 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.

Important

Après modification, vous devez redémarrer le service pour charger la nouvelle configuration avec la commande : `safekit webserver restart` (voir section 9.2 page 145).

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 section 10.6.2 page 170 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 page 195 pour plus de details).

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`.

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`

10.6.2 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 page 183 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

10.6.3 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 section 11 page 177.

Pour revenir à la configuration HTTP si celle-ci a été changée pour HTTPS, voir 11.2.1.1 page 180.

10.6.4 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.

10.7 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 14 page 269.

Pour un exemple complet avec le module de démonstration `notification.safe`, voir 15.14 page 292.

10.8 Surveillance SNMP

SafeKit peut être surveillé via SNMP. Depuis la version 8, les implémentations pour Windows et Linux différent : En Windows, SafeKit utilise son propre agent snmp, alors qu'en Linux, l'agent snmp du systeme est utilisé.

10.8.1 Surveillance SNMP en Windows

Pour utiliser l'agent SNMP de SafeKit, `safeagent`, 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 `safeagent` ("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/safekit firewallcfg add
```

3. le démarrer avec la commande :

`safekit safeagent [start | stop | restart | check]`	Contrôle le démarrage/arrêt du service `safeagent` qui met en œuvre un agent SNMP SafeKit.

La configuration du service `safeagent` 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`.

10.8.2 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-feux 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

10.8.3 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 par exemple 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.

10.9 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 web@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.

10.10 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
```

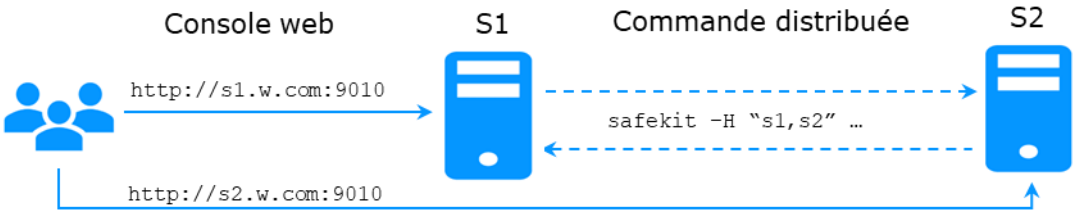
11.Sécurisation du service web de SafeKit

11.1 Vue générale

Le service web de SafeKit est principalement utilisé par :

- La console web (voir section 3 page 37)
- L'interface en ligne des commandes distribuées sur le cluster (voir section 9.1 page 143)

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 11.2.1 page 179

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 11.2.2 page 181
A base de fichiers	✓ HTTP ✓ HTTPS Pour configurer HTTPS avec : ➪ la PKI SafeKit, voir 11.3.1 page 183 ➪ une PKI externe, voir 11.3.2 page 191	✓ 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 11.4.1 page 196
LDAP/AD	✓ HTTP ✓ HTTPS	✓ Authentification à base de serveur LDAP/AD ✓ Gestion de rôles facultative

	Pour configurer HTTPS avec : ➪ la PKI SafeKit, voir 11.3.1 page 183 ➪ une PKI externe, voir 11.3.2 page 191	Pour la configuration, voir 11.4.2 page 198
OpenID Connect	✓ HTTP ✓ HTTPS Pour configurer HTTPS avec : ➪ la PKI SafeKit, voir 11.3.1 page 183 ➪ une PKI externe, voir 11.3.2 page 191	✓ Authentification à base de serveur OpenID Connect ✓ Gestion de rôles facultative Pour la configuration, voir 11.4.3 page 201

Important

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 11.2.1 page 179.

Elle peut aussi être remplacée par la configuration minimale décrite en 11.2.2 page 181 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 11.4.1 page 196
- ✓ pour passer en HTTPS, avec :
  - ➪ la PKI SafeKit, décrit en 11.3.1 page 183
  - ➪ une PKI externe, décrit en 11.3.2 page 191

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 11.4.1 page 196.

Si vous désirez revenir au mode HTTP, sur tous les serveurs SafeKit :

- Supprimer le fichier : `SAFE/web/conf/ssl/httpd.webconsolessl.conf`
- 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`.

`webservercfg -passwd <pwd>`	Sur S1 et S2 :
	➪ en Windows, ouvrir une console PowerShell en tant qu'administrateur et exécuter (`SAFE=C:\safekit` si `%SYSTEMDRIVE%`=C:)
	`SAFE/private/bin/webservercfg.ps1 -passwd <pwd>`
	➪ en Linux, ouvrir une console Shell en tant que root et exécuter (`SAFE=/opt/safekit`)
	`SAFE/private/bin/webservercfg -passwd <pwd>`
	Vous devez affecter le même mot de passe sur tous les nœuds.

Important

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 11.4.1.1 page 196).

  En cas d'échec de l'authentification dans la console, vous devrez peut-être réinitialiser le mot de passe. Pour cela, exécutez à nouveau `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 `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 : ➪ éditer le fichier `SAFE/web/conf/httpd.conf` ➪ commenter `usefile`, `useldap` et `useopenid` `#Define usefile`

	… #Define useldap … #Define useopenid ➪ 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 : ➪ détruire le fichier `SAFE/web/conf/ssl/httpd.webconsolessl.conf`
	Sur S1 et S2 : ➪ 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 :

CA	Le certificat de l'Autorité de Certification CA utilisée pour générer les certificats serveur de S1 et S2
S1 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 :

- 11.3.1 « Configuration HTTPS avec la PKI SafeKit » page 183

  Aller à cette section pour une configuration rapide de HTTPS avec l'autorité de certification « privée » de SafeKit

- 11.3.2 «Configuration HTTPS avec une PKI externe» page 191

  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 page 195.

11.3.1 Configuration HTTPS avec la PKI SafeKit

Important

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`).

Note

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.

Important

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 `safekit 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
     ```

Important

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

Note

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
     ```

Important

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
   ```

Important

Si nécessaire, définissez les variables d'environnement `HTTPS_PROXY` and `HTTP_PROXY` avec les valeurs adéquates.

Note

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 :

- copier `SAFE/web/conf/httpd.webconsolessl.conf` dans `SAFE/web/conf/ssl/httpd.webconsolessl.conf`
- En Linux exécuter :

  ```
  chown safekit:safekit SAFE/web/conf/ssl/httpd.webconsolessl.conf
  chmod 0440 SAFE/web/conf/ssl/httpd.webconsolessl.conf
  ```

- 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

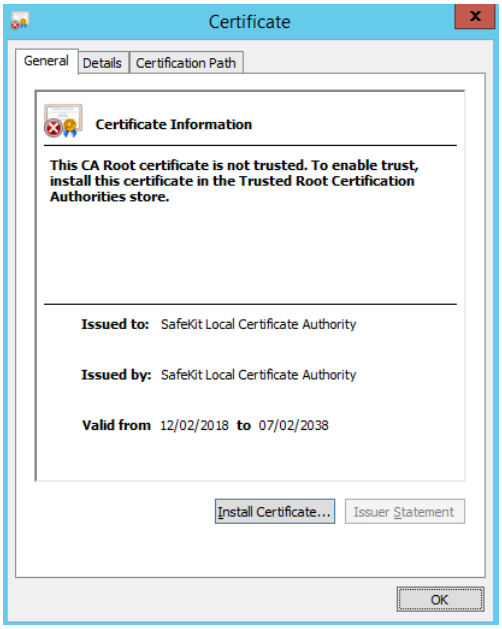
Une fois le service Web SafeKit configuré en HTTPS, les communications réseau peuvent être ouvertes en configurant le pare-feu comme décrit en 10.3 page 160.

11.3.1.7 Utiliser la console web SafeKit en HTTPS

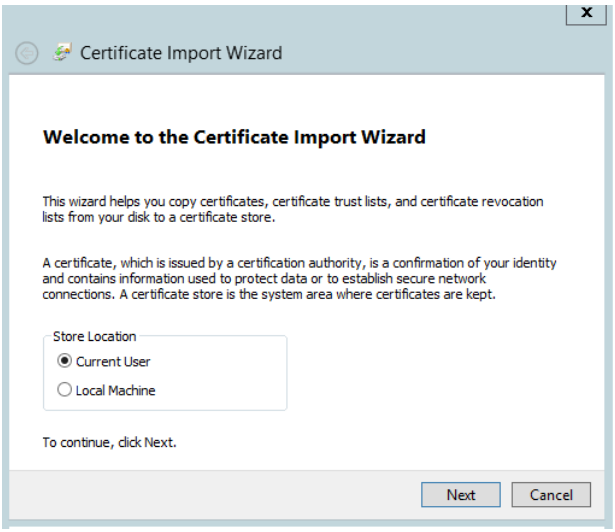
Tant que le certificat de l'autorité de certification n'a pas été importé, le navigateur émet des alertes de sécurité lorsque l'utilisateur se connecte à la console web avec son certificat client. Si l'importation n'a pas déjà été faite, appliquez la procédure ci-dessous en Windows :

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`

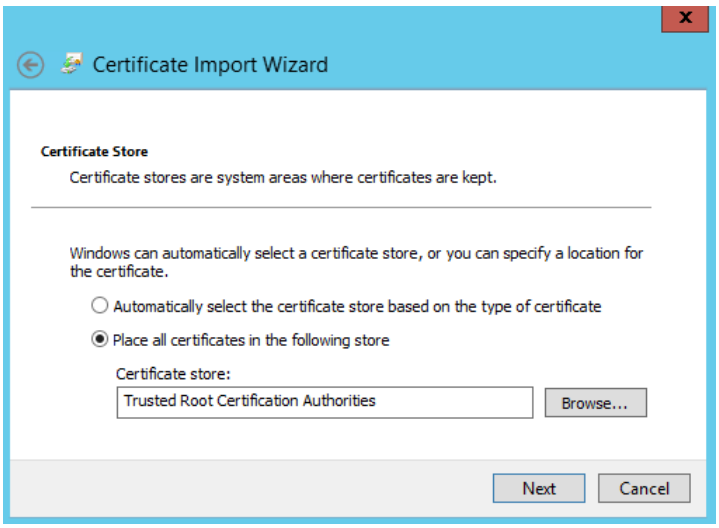
3. Cliquer sur le fichier `cacert.crt` téléchargé pour ouvrir la fenêtre Certificate. Cliquer ensuite sur le bouton Install Certificate



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.

- Se connecter en tant qu'administrateur/root et ouvrir une fenêtre d'invite de commandes
- Aller dans le répertoire `SAFE/web/bin`
- 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 :

- Supprimer le répertoire `web/conf/ca` sur tous les serveurs, y compris le serveur CA
- Supprimer les certificats des magasins des stations de travail des utilisateurs
- Réappliquer complètement les procédures décrites en section 11.3 page 183

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.

Note

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.

CA	Les certificats serveur de S1 et S2 doivent être signés par l'autorité de certification CA
S1	Le certificat serveur de S1 permettant de l'authentifier
S2	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` 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 Voir l'exemple dans 11.3.2.1.3 page 193.
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 s1.crt s1.key	Sur S1 : ➪ copier `s1.crt` dans `SAFE/web/conf/server.crt` ➪ copier `s1.key` dans `SAFE/web/conf/server.key`
S2 s2.crt s2.key	Sur S2 : ➪ copier `s2.crt` dans `SAFE/web/conf/server.crt` ➪ copier `s2.key` dans `SAFE/web/conf/server.key`

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
```

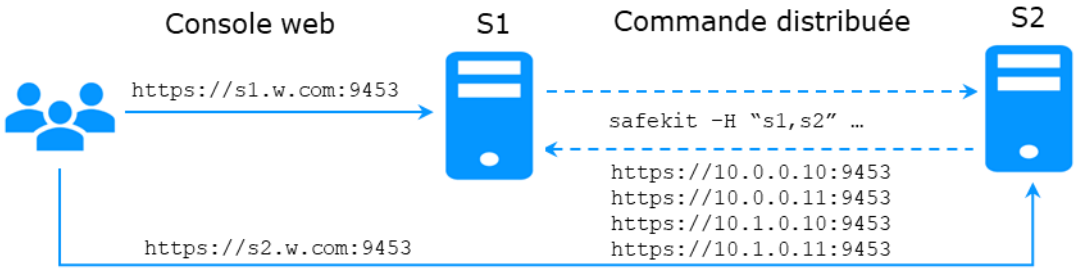
Cette commande retourne en échec si une erreur est détectée.

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.

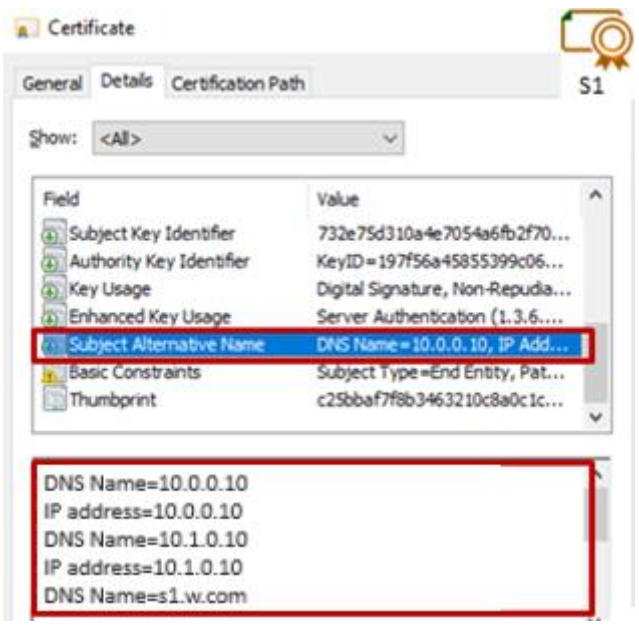
Pour vérifier que cela est bien le cas :

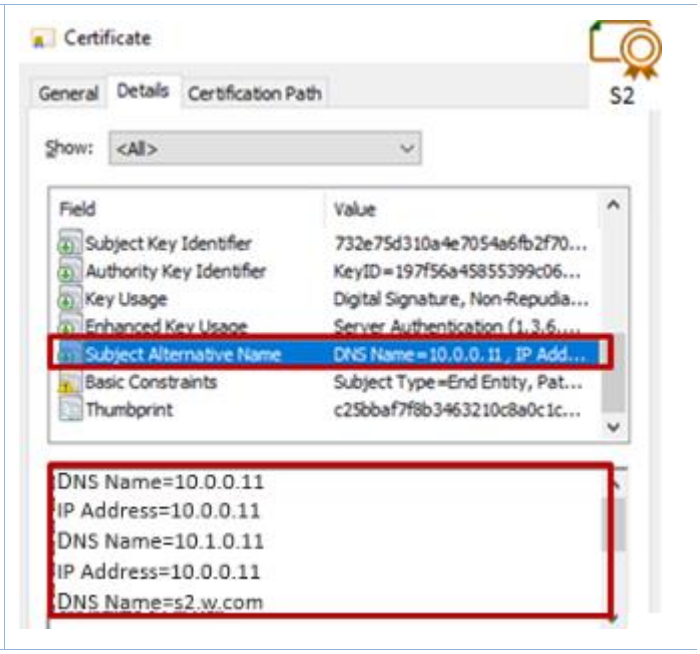
1. Copier le fichier `.crt` (ou `.cer`) sur une station de travail Windows
2. Double cliquer sur le fichier pour l'ouvrir avec Extension noyau de chiffrement
3. Cliquer sur l'onglet Détails
4. Vérifier le contenu du champ `Autre nom de l'objet` (`Subject Alternative Name`)

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 :

CA `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	S1 S2 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 7.18. page 129.

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) :

CA `cacert.crt`	Sur S1 et S2 : ➪ copier `cacert.crt` dans `SAFE/web/conf/cacert.crt` ➪ 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:

- copier `SAFE/web/conf/httpd.webconsolessl.conf` dans `SAFE/web/conf/ssl/httpd.webconsolessl.conf`
- En Linux exécuter :

  `chown safekit:safekit SAFE/web/conf/ssl/httpd.webconsolessl.conf`

  `chmod 0440 SAFE/web/conf/ssl/httpd.webconsolessl.conf`
- 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 `safekit 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 S1 et S2 : ➪ exécuter `safekit firewallcfg add`

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 10.3 page 160.

11.4 Configuration de l'authentification utilisateur

Mettez en œuvre l'une des méthodes suivantes pour l'authentification utilisateur :

- 11.4.1 « Configuration l'authentification à base de fichier » page 196
- 11.4.2 « Configuration de l'authentification à base de serveur LDAP/AD » page 198
- 11.4.3 « Configuration de l'authentification à base de serveur OpenID Connect » page 201

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 :

user.conf	Contrôle d'accès à partir du fichier des utilisateurs
Admin Control Monitor group.conf	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).

Note

Pendant l'initialisation de la configuration par défaut, décrite dans 11.2.1 page 179, 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.

- 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` :

user.conf	`admin:$2y$05$oPquL6Z2Y78QcXpHIako.O58Z6lWfa5A86XD.eCbEnbRcguJln9Ce`
	`manager``:$apr1$U2GLivF5$x39WKmSpq6BGmLybESgNV1`
	`operator1:$apr1$DetdwaZz$hy5pQzpUlPny3qsXrIS/z1`
	`operator2:$apr1$ICiZv2ru$wRkc3BclBhXzc/4llofoc1`

- 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.

Note

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` :

Admin Control Monitor group.conf	`Admin : admin`
	`Control : manager`
	`Monitor : operator1 operator2`

Note

Si vous activez la gestion de rôle, vous devez insérer l'utilisateur `admin` dans `group.conf`. Sinon, cet utilisateur ne sera plus opérationnel.

- 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 : ➪ copier `user.conf` dans `SAFE/web/conf/user.conf`
Admin Control Monitor group.conf	Sur S1 et S2, si les groupes sont définis : ➪ copier `group.conf` dans `SAFE/web/conf/group.conf`

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) :

	Sur S1 et S2 : ➪ éditer le fichier `SAFE/web/conf/httpd.conf` ➪ si nécessaire, décommenter `usefile` `Define usefile`
	Sur S1 et S2 : ➪ 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`.
1. 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.
2. 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

ADMIN CONTROL MONITOR	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.

Note

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 : ➪ exécuter `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.


httpd.conf
	Sur S1 et S2 :
	➪ éditer le fichier `SAFE/web/conf/httpd.conf`
	➪ décommenter `useldap`
	`Define useldap`
	➪ 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

Note

`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 :

- éditer le fichier `SAFE/web/conf/httpd.conf`
- 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 :

- 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

L'authentification OpenID connect s'appuie sur le module Apache `mod_auth_openidc`. Elle peut être appliquée en HTTP ou 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.
ADMIN CONTROL MONITOR	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.

Note

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 valeur **Erreur ! Référence de lien hypertexte non valide.** du serveur>:9453/openid ou **Erreur ! Référence de lien hypertexte non valide.** du serveur>:9010/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 : ➪ exécuter `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.


httpd.conf
	Sur S1 et S2 :
	➪ éditer le fichier `SAFE/web/conf/httpd.conf`
	➪ décommenter `useopenid`
	`Define useopenid`
	➪ 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 :
	➪ éditer le fichier `SAFE/web/conf/httpd.conf`
	➪ 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 : ➪ 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). 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

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 type de réseau est aussi utilisé pour l'exécution des commandes distribuées.

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.

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 section 13.3 page 215)
- dans la section lan pour un module ferme (voir section 13.4 page 217)

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. Le réseau nommé `private` peut être dédié au flux de la réplication de fichiers.

```
<cluster>
   <lans>
      <lan name="default">
         <node name="node1" addr="192.168.1.67"/>
         <node name="node2" addr="192.168.1.68"/>
         <node name="node3" addr="192.168.1.69"/>
         <node name="node4" addr="192.168.1.70"/>
      </lan>
      <lan name="repli">
         <node name="node1" addr="10.0.0.1"/>
         <node name="node2" addr="10.0.0.2"/>
         <node name="node3" addr="10.0.0.3"/>
         <node name="node4" addr="10.0.0.4"/>
      </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.

Note

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.

Note

- ✓ 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

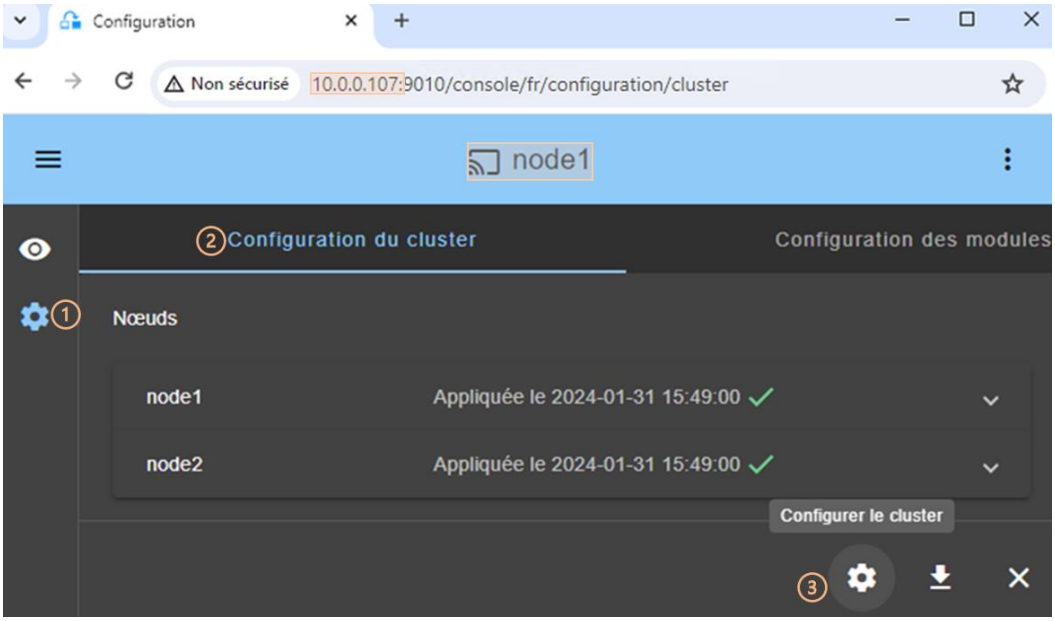
Ouvrir l'assistant de configuration du cluster

✓ 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 page 40.

12.2.2 Configuration en ligne de commande

Pour la description complète des commandes, se référer à la section 9.3 page 146.

Les commandes pour configurer le cluster avec une nouvelle clé de chiffrement sont :

1. `safekit cluster config [<filepath>]`

   où `filepath` est le chemin d'accès au nouveau `cluster.xml`

   quand `filepath` n'est pas passé en argument, la configuration courante est conservée et seule la clé d'encryption est régénérée

2. `safekit -H "*" -G`

   la configuration local, définie dans le fichier `cluster.xml`, est appliquée sur les nœuds du cluster

Les commandes pour reconfigurer sans clé de chiffrement sont :

1. `safekit cluster delkey`
2. `safekit -H "*" -G`

Les commandes pour régénérer des clés de chiffrement et les prendre en compte sont :

1. `safekit cluster genkey`
2. `safekit -H "*" -G`

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 9.3 page 146). 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. Appliquer la nouvelle configuration, modifiée sur `node1`, sur tous les nœuds avec (remplacer `node1`, `node2` par le nom des nœuds et `AM` par le nom du module) :

- ✓ la console web en naviguant sur ⚙Configuration/Configuration des modules/ ⚙Configurer le module/
- ✓ ou en entrant directement l'URI /console/fr/configuration/modules/AM/config/
- ✓ ou avec la commande `safekit config -H "node1,node2" -m AM` exécutée sur `node1`

Exemple de `userconfig.xml`:

```
<safe>
  <!-- Insert below <macro> <service> tags -->
</safe>
```

Important

Avec la console web, le module doit être arrêté avant d'appliquer la configuration.

Ave 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> tag)

13.1.1 <macro> Exemple

```
<macro name="ADDR1" value="aa.bb.com"/>
```

Un exemple d'utilisation de macro est donné dans 15.3 page 279.

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.
/>	

Note

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.

13.2 Module ferme ou miroir (<service> tag)

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>
```

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>
```

Des exemples de définition de <service> sont donnés pour un module miroir dans 15.1 page 276 et pour un module ferme dans 15.2 page 277.

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"`	`mirror` pour un module miroir. Le protocole de synchronisation entre les 2 serveurs est défini en 13.3 page 215. Voir le module applicatif mirror.safe en tant qu'exemple. `farm` pour un module ferme. Le protocole de synchronisation entre les serveurs est défini en 13.4 page 217. Voir le module applicatif farm.safe en tant qu'exemple. `light` pour une configuration sur un seul serveur avec une détection d'erreur logicielle et un redémarrage local seulement.
`[boot= "on"| "off"| "auto"| "ignore"]`	Si `on`, le module est automatiquement démarré au boot de la machine. Si `off`, le module n'est pas démarré au boot de la machine. Si `auto`, le module est démarré automatiquement au boot de la machine s'il était démarré avant le reboot. 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. Si `on`, reprise automatique sur le serveur secondaire lorsque le serveur primaire est arrêté. Si `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 page 102. 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. Avec `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. Avec `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 1.5.1 page 20) ou N-1 (voir section 1.5.2 page 21). Avec `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 maximum de `restart` ou `stopstart` appelés par les détecteurs d'erreur avant d'arrêter localement SafeKit. Ce compteur est réinitialisé à sa valeur initiale à l'expiration du timeout `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` décrémente le compteur, alors qu'une commande manuelle sans cette option ne décrémente pas le compteur. Pour plus d'informations, voir section 13.18.4 page 267. Note: 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 Note 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` Note La valeur de cet attribut peut être modifiée dynamiquement.

13.3 Heartbeats (<heart>, <heartbeat > tags)

Les heartbeats doivent être utilisés seulement avec les modules miroirs. La topologie d'une ferme est décrite dans la section 13.4 page 217.

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 splitbrain.

13.3.1 <heart> Exemple

```
<heart>
   <heartbeat name="default" ident="Hb1" />
   <heartbeat name="net2" ident="Hb2" />
</heart>
```

Un exemple de configuration de heartbeats avec de multiples voies est donné en 15.4 page 280.

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"]
 >
  [<!-- syntaxe pour SafeKit < 7.2 -->
   <server addr="IP1_address" />
   <server addr="IP2_address" />
  ]
</hearbeat>
 …
</heart>
```

Note

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 page 205).

[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 page 266). Si l'attribut ident n'est pas défini la valeur de l'attribut name est utilisée. Important: 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" />	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. Important: 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> tags)

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).

13.4.1 <farm> Exemple

```
<farm>
   <lan name="default" />
   <lan name="net2" />
</farm>
```

Pour des exemples de configuration <farm>, voir section 15.5 page 280.

13.4.2 <farm> Syntaxe

```
<farm [port="xxxx"]>
   <lan name="network>
     [<!-- syntaxe pour SafeKit < 7.2 -->
```

```
      <node name="server1" addr="IP1_address"|"IP1_name" />
      <node name="server2" addr="IP2_address"|"IP2_name" />
     ]
   </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 page 205).
`< 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. Important: 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> tag)

Important

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 une architecture 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 en 15.2 page 277.

13.5.2 <vip> Exemple dans une architecture 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 en 15.1 page 276.

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="integer" />
        …
     </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>
```

Note

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 page 258.
[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"	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 13.5.7.3 page 226 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 13.5.7.2 page 225

	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 page 259.
[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 réel 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 page 259.
[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

Pour des exemples de load balancing, voir section 15.5 page 280.

A utiliser avec un module ferme

<loadbalancing_list>	
<group	Définition d'un groupe de load balancing. Mettre autant de sections qu'il y a de groupes : un exemple est donné en 15.5.3 page 282.
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 page 205).
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 13.5.7.4 page 226.
</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. Voir l'exemple en 15.5.1 page 280. filter="on_ipid" La règle de load balancing est réalisée sur l'ip_id en entrée. Utile pour UDP seulement (voir l'exemple en 15.5.2 page 281).

13.5.7 <vip> Description

13.5.7.1 <vip> prérequis

Voir les prérequis réseau décrits en 2.3.2 page 31.

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.

Pour tester une adresse MAC virtuelle sur votre réseau, faire d'abord le test de compatibilité décrit en 4.3.7 page 84.

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 4.3.5 page 82.

13.6 Réplication de fichiers (<rfs>, <replicated> tags)

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 13.6.4.2 page 236.

Sur Windows, il est vivement recommandé d'activer le journal USN sur le lecteur contenant le répertoire répliqué, comme décrit en 13.6.4.3 page 238.

Important

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 l'exemple de flux de réplication dédié décrit en 15.4 page 280.

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"]
>

  [<flow name="network" >
    [<!-- syntaxe pour SafeKit < 7.2 -->
     <server addr="IP_address_1" />
```

```
     <server addr="IP_address_2" />
    ]
  </flow>]

 <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/dir`*`suffix`*". Le répertoire **/a/dir** est créé et c'est : ➪ un point de montage vers `/a/dir`*`suffix`* lorsque le module est démarré ➪ un lien vers "`/a/dir`*`suffix`*" 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`.

	Important 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` Important **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. Positionner `async="none"` assure plus de sécurité : les opérations d'écriture répliquées sont mises sur disque avant d'envoyer les acquittements au serveur primaire. Avec `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` Note 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. 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"`. 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 Note: 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`

	Note: La valeur de cet attribut peut être modifiée dynamiquement.
[namespacepolicy="0"|"1"|"3"|"4"]	En Windows, avec l'option `namespacepolicy="1"`, la réintégration par zones ne peut être assurée après l'arrêt propre du module, puis reboot du serveur. Pour que ce cas soit supporté en Windows, il faut positionner `namespacepolicy="3"`. 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 : ➪ l'USN journal associé au volume a été détruit/recréé par l'administrateur ➪ une discontinuité dans le journal est détectée 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. Utiliser `namespacepolicy="0"` pour désactiver la synchronisation par zones sur Windows ou Linux. 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 Note: 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`

	Note 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`: • Lors de la 1ère réintégration • Lors d'un démarrage en réintégration complète (`safekit second fullsync`) • 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) Avec l'option `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. Lorsque `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 externe (voir le module Linux DRBD.safe pour plus d'information) 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. Note: 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"]`	Quand cet attribut est <=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. Quand cet attribut est >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 13.6.4.10 page 245. 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 * * *" Note: Voir la documentation de crontab en Unix et de schtasks.exe en Windows, pour une description complète de la syntaxe Important: 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 page 215). 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 13.18.5 page 267). Note: 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 page 205). 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. Important: 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>`
	
	Important: 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

	Important: 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 2.2.4 page 30.

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.

Important

C'est la même procédure pour tous les points de montage qui doivent être répliqués.

- démonter les systèmes de fichiers en exécutant :

  `umount /var/lib/pgsql/var`

  `umount /var/lib/pgsql/data`

- configurer le module en exécutant :

  `/opt/safekit/safekit config -m postgresql`

  La configuration se termine avec succès.

- 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`

- 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..`

- monter les systèmes de fichiers en exécutant :

  `mount /var/lib/pgsql/var_For_SafeKit_Replication`

  `mount /var/lib/pgsql/data_For_SafeKit_Replication`

Important

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 ...

Note

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 :

- démonter les systèmes de fichiers en exécutant :

  `umount /var/lib/pgsql/var_For_SafeKit_Replication`

```
umount /var/lib/pgsql/data_For_SafeKit_Replication
```

- déconfigurer le module en exécutant `/opt/safekit/safekit deconfig -m postgresql`
- éditer `/etc/fstab` pour y restaurer son état initial
- 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\paramete
rs] "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.

Note

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 page 228). 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.

Important

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.

Important

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.

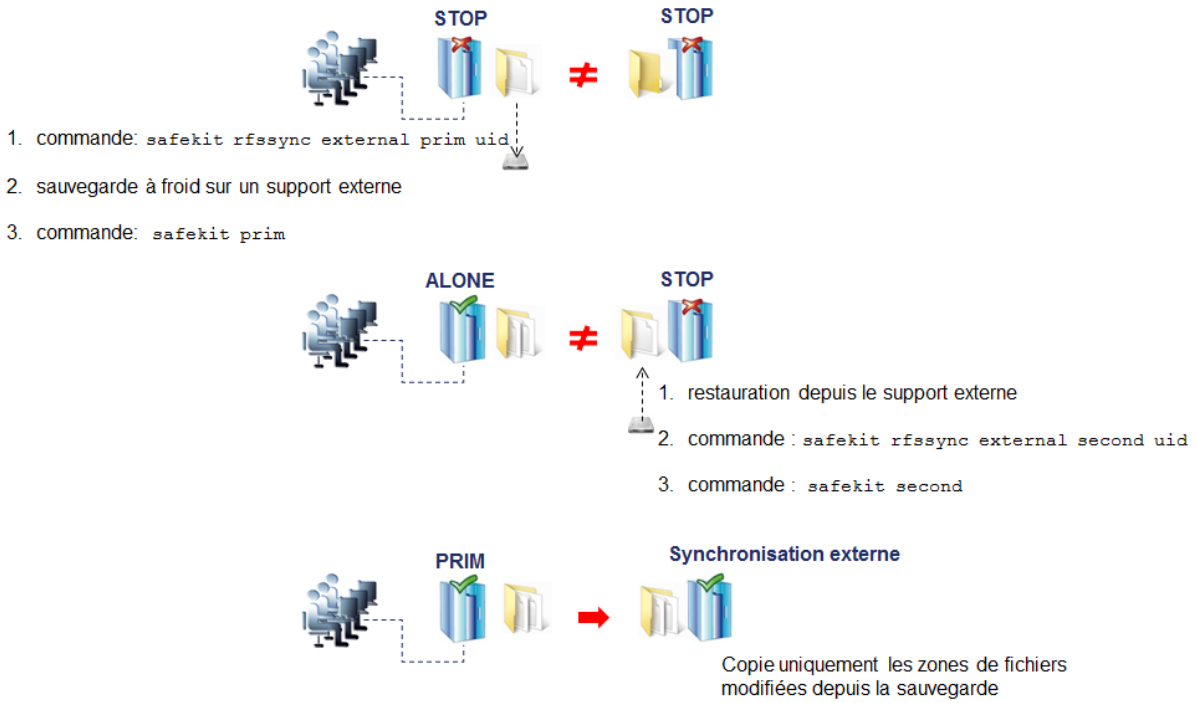
Sur le nœud principal, il suffit d'effectuer une sauvegarde des répertoires répliqués et de passer la politique synchronisation au mode externe. La sauvegarde est transportée (en utilisant un lecteur externe par exemple) et restaurée sur le nœud secondaire, qui est aussi configuré pour effectuer une 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 :

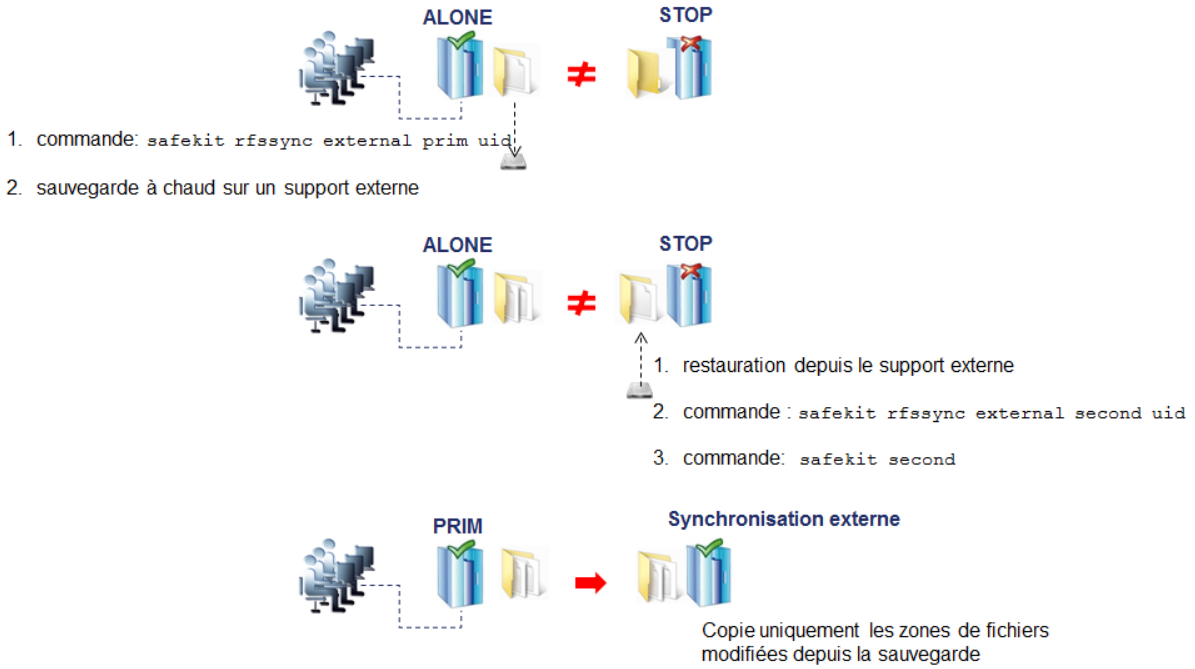
- le rôle du nœud (`prim | second`)
- un identificateur unique (`uid`)

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. Important: 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 page 241). 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 page 228).

13.7 Activer les scripts du module (<user>, <var> tags)

Cette section décrit uniquement les options de configuration du tag `<user>`. Pour une description complète des scripts, voir la section 14 page 269.

13.7.1 <user> Exemple

```
<user logging="userlog" >
      <var name="VARENV" value="V1" />
</user>
```

Voir un exemple en 15.1 page 276.

13.7.2 <user> Syntaxe

```
<user
   [nicestoptimeout="300"]
   [forcestoptimeout="300"]
   [logging="userlog"|"none"]
   [userlogsize="2048"]
 >
      <var name="ENVIRONMENT_VARIABLE_1" value="VALUE_1" />
      …
</user>
```

Note

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>_<scr name>.ulogoù AM est le nom du module (SAFEVAR=C:\safekit\var s Windows et /var/safekit sur LINUX). Note: Depuis SafeKit 7.4.0.19, l'extension du nom de fichier du journal des scripts a changé. Le fichier s'appelle dorénavant userlog_<year>_<month>_<day>T<time>_<script name>.ulogau lieu de userlog.AM 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> tags)

13.8.1 <vhost> Exemple

```
<vhost>
  <virtualhostname name="vhostname" envfile="vhostenv" />
</vhost>
```

Voir l'exemple en 15.6 page 283.

13.8.2 <vhost> Syntaxe

```
<vhost>
```

```
  <virtualhostname
     name="virtual_hostname"
     envfile="path_of_a_file"
    [when="prim"|"second"|"both"]
  />
</vhost>
```

Note

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.

Pour un exemple complet, voir 15.6 page 283.

13.9 Détection de la mort de processus ou de services (<errd>, <proc> tags)

Important

La section <errd> nécessite d'avoir défini la section <user/>

13.9.1 <errd> Exemple

13.9.1.1 Surveillance de processus

Linux and 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>
```

Voir l'exemple en 15.7 page 285.

13.9.1.2 Surveillance de service

myservice est le nom du service Windows (pour safekit > 7.3) ou du service systemd Linux (pour safekit > 7.4.0.19) à surveiller :

```
<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" **Ou** name="command_name" nameregex="regular expression on the command name"	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" et sur Windows name="notepad.exe". Important: En Windows. Le nom est automatiquement converti en minuscule. Pour retrouver le nom des commandes associées aux processus, voir les commandes décrites en 13.9.4 page 252. *Linux uniquement* nameregex est une expression régulière sur le nom de la commande pour sélectionner le processus à surveiller. name est le nom de la ressource associée au processus surveillé. Important: Comme les expressions régulières sont définies dans le fichier XML userconfig.xml, certains caractères ne peuvent pas être utilisés '<' ou '>'.

Ou name="service_name" service="yes"	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 name est le nom du service à surveiller. C'est aussi le nom de la ressource associée au service surveillé. Au maximum, 63 caractères. Exemples : 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
class= "prim"| "both"| "pre"| "second"| "sec"| "othername"	Le processus appartient à une classe. La surveillance démarre avec la commande safekit errd enable classname -m AM. Les fonctions enable/disable des classes prim, both, pre, second, sec sont automatiques et faites par SafeKit 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 14 page 269. Avec un autre nom de classe, les fonctions enable/disable doivent être faites 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.
	⇨ 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" …`

	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 Pour retrouver la liste des arguments d'un processus, utiliser les commandes décrites en 13.9.4 page 252. Important Comme les expressions régulières sont définies dans le fichier XML `userconfig.xml`, certains caractères ne peuvent pas être utilisés '`<`' ou '`>`'.
`atleast="1"`	Nombre minimum de processus qui doivent s'exécuter. Si le minimum est atteint, SafeKit déclenche l'action. Exemple: `name="oracle" argregex=".*db1.*" atleast="1"` signifie qu'une action est déclenchée si 0 processus s'exécute sur l'instance `oracle/db1`. S'il est positionné à `-1`, ce critère 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. `noaction` réalise juste un logging de message, `restart` un redémarrage local et `stopstart` un basculement. Les commandes restart/stopstart incrémente le compteur `maxloop` pour vérifier que l'on n'est pas sur une faute reproductible. Pour la description de `maxloop`, voir 13.2 page 212. 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 1 quand la limite est atteinte.

	Voir l'exemple décrit en 15.7 page 285. 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 suspend -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.
`safekit errd resume -m AM`	Redémarre la surveillance des processus du module `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> tags)

SafeKit apporte des checkers avec des règles de failover par défaut (voir section 13.18.5 page 267). Les checkers sont :

- 13.11 « TCP checker (<tcp> tags) » page 256.
- 13.12 « Ping checker (<ping> tags) » page 257.
- 13.13 « Interface checker (<intf> tags) » page 258.
- 13.14 « IP checker (<ip> tags) » page 259
- 13.15 « Custom checker (<custom> tags) » page 261.
- 13.16 « Module checker (<module> tags) » page 263
- 13.17 « Splitbrain checker (<splitbrain> tag) » page 264

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.

13.11 TCP checker (<tcp> tags)

Important

Par défaut, un checker <tcp> réalise un redémarrage local du module lorsque le service TCP est down.

13.11.1 <tcp> Exemple

```
<check>
  <tcp ident="R1test" when="prim" >
    <to addr="R1" port="80"/>
  </tcp>
</check>
```

Important

Insérer le tag <tcp> dans la section <check> si celle-ci est déjà définie.

Voir l'exemple en 15.8 page 287.

13.11.2 <tcp> Syntaxe

```
  <tcp
    ident="tcp_checker_name"
    when="prim|second|both|pre"
  >
    <to
     addr="IP_address" or "name_to_check"
     port="TCP_port_to_check"
     [interval="10"]
     [timeout="5"]
     />
  </tcp>
```

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.
`when="prim|second|both"`	Utiliser cette valeur pour tester un service TCP interne à l'application Suivant cette valeur, le checker est démarré/arrêté après/avant les scripts `start_prim/stop_prim`, `start_second/stop_second`, `start_both/stop_both`. Action en cas de défaillance: `restart` du module (voir les règles de failover par défaut décrites en 13.18.5 page 267). Chaque restart incrémente le compteur `maxloop` (voir section 13.2.3 page 213).

when="pre"	Utiliser cette valeur pour tester un service TCP externe à l'application Le checker est démarré/arrêté après/avant les scripts prestart/poststop. Vous devez ajouter une règle de failover spéciale pour un tel checker. Typiquement: external_tcp_service: if (tcp.tcp_checker_name == down) then wait(); Cette règle réalise un stopwait et met le module dans l'état WAIT tant que le service TCP externe ne répond pas (pour plus d'information, voir 13.18 page 266). Chaque stopwait incrémente le compteur maxloop (voir section 13.2.3 page 213).
<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> tags)

Par défaut, un <ping> checker met le module dans l'état WAIT et attend que ping redevienne up.

13.12.1 <ping> Exemple

```
<check>
  <ping ident="testR2" >
    <to addr="R2"/>
  </ping>
</check>
```

Insérer le tag <ping> dans la section <check> si celle-ci est déjà définie.

Voir l'exemple en 15.9 page 287.

13.12.2 <ping> Syntaxe

```
  <ping
    ident="ping_checker_name"
    [when="pre"]
```

```
  >
    <to
     addr="IP_address" or "name_to_check"
     [interval="10"]
     [timeout="5"]
     />
  </ping>
```

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
`[when="pre"]`	Valeur par défaut Le checker est démarré/arrêté après/avant les scripts `prestart`/`poststop`. La règle de failover par défaut réalise un stopwait qui met le module dans l'état `WAIT` tant que le ping ne répond pas (pour plus d'informations, voir 13.18 page 266). Chaque stopwait incrémente le compteur `maxloop` (voir 13.2.3 page 213).
`<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> tags)

Par défaut, un checker <intf> arrête le module et attend que l'interface réseau soit up.

13.13.1 <intf> Exemple

```
<check>
  <intf ident="test_eth0">
    <to local_addr="192.168.1.10"/>
  </intf>
</check>
```

Important

Insérer le tag <intf> dans la section <check> si celle-ci est déjà définie.

Voir l'exemple en 15.10 page 287.

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`	Note: Les sections <intf> sont automatiquement générées lorsque `<interface check="on">` est positionné (voir section 13.5 page 219).
`ident="intf_checker_name"`	Le nom de l'interface checker (adresse IP du réseau).
`[when="pre"]`	Valeur par défaut Le checker est démarré/arrêté après/avant les scripts `prestart`/`poststop`. La règle de failover par défaut réalise un stopwait qui met le module dans l'état `WAIT` tant que le ping ne répond pas (pour plus d'informations, voir 13.18 page 266). Chaque stopwait incrémente le compteur `maxloop` (voir 13.2.3 page 213).
`<to local_addr="IP_@ />`	Adresse IP associée à l'interface à tester. Adresse IPv4 ou IPv6.
`</intf>`	

13.14 IP checker (<ip> tags)

En Windows et en Linux, ce checker vérifie qu'une adresse IP est bien configurée localement ; en Windows, il détecte en plus les conflits sur cette adresse.

Important

Par défaut, un checker <ip> exécute un stopstart local du module lorsque l'adresse testée est down.

13.14.1 <ip> Exemple

```
<check>
  <ip ident="ip_check" >
    <to  addr="192.168.1.10" />
  </ip>
</check>
```

Important

Insérer le tag <ip> dans la section <check> si celle-ci est déjà définie.

Voir l'exemple en 15.11 page 288.

13.14.2 <ip> Syntaxe

```
  <ip
    ident="ip_checker_name"
    [when="prim"]
  >
    <to
     addr="IP_address" or "name_to_check"
     [interval="10"]
     />
  </ip>
```

13.14.3 <ip> Attributs

`<ip`	Mettre autant de section <ip> qu'il y a de checkers ip.
`ident="ip_checker_name"`	Nom du checker ip (dont l'état est affiché par la commande `safekit state -v -m AM)`. Chaque checker doit avoir un nom différent.
`[when="prim"]`	Valeur par défaut Le checker est démarré/arrêté après/avant les scripts `prestart`/`poststop`. La règle de failover par défaut réalise un stopstart du module (pour plus d'informations, voir 13.18 page 266). Chaque stopstart incrémente le compteur `maxloop` (voir 13.2.3 page 213).
`<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> tags)

Un checker customisé est un programme (script ou autre) 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 une ressource à "up" ou "down". Puis, une règle de failover dans `userconfig.xml` décide l'action à exécuter lorsque la ressource est down

13.15.1 <custom> Exemple

```
<check>
  <custom ident="AppChecker" when="prim" exec="mychecker" action="restart"/>
</check>
```

Important

Insérer le tag <custom> dans la section <check> si celle-ci est déjà définie. Définir en plus le checker customisé.

Pour des exemples complets, voir les sections 15.12 page 289.

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"
/>
```

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 Un checker customisé positionne sa ressource avec la commande `safekit set -r custom.custom_checker_name -v up|down`.
`when="pre"`	Le checker est démarré/arrêté après/avant les scripts `prestart`/`poststop`. Vous devez spécifier en plus l'attribut `action="wait"`. Elle a pour effet d'exécuter un `stopwait` et de mettre le module dans l'état `WAIT` tant que la ressource est `down`. Noter que l'état initial de la ressource est `init` et que la failover machine reste dans l'état `WAIT` tant que ressource n'est pas positionnée à up (pour plus d'information, voir 13.18 page 266). Chaque `stopwait` incrémente le compteur `maxloop` (voir 13.2.3 page 213).

	Important: Dans SafeKit < 8, l'action était configurée en définissant une règle de failover dans le tag `<failover>`. Par exemple : `wait_custom_checker: if (custom.custom_checker_name == down) then wait();`
`when="prim"|"second"|"both"`	Suivant cette valeur, le checker est démarré/arrêté après/avant les scripts `start_prim/stop_prim`, `start_second/stop_second`, `start_both/stop_both`. Vous devez spécifier en plus l'attribut `action="stop|stopstart|restart"`. Pour plus d'information, voir 13.18 page 266. A chaque détection d'erreur, le compteur `maxloop` est incrémenté (voir 13.2.3 page 213). Important: Dans SafeKit < 8, l'action était configurée en définissant une règle de failover dans le tag `<failover>`. Par exemple: `restart_custom_checker: if (custom.custom_checker_name == down) then restart();`
`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 10.1 page 157). 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.
`action="wait|stop|stopstart|restart"`	Défini la règle de failover associée qui effectue l'action spécifiée si la ressource est à l'état `down`. La règle de failover est nommée : `c_<ident value>`.

13.16 Module checker (<module> tags)

Le checker de module teste la disponibilité d'un autre module. Le checker est démarré/arrêté après/avant les scripts `prestart`/`poststop`. Lorsque le checker de module détecte qu'un module externe est down, la failover machine réalise un `stopwait` et met le module local en `WAIT` jusqu'à ce que le module externe soit de nouveau détecté up. 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 un failover).

Chaque `stopwait` ou stopstart incrémente le compteur `maxloop` (voir 13.2.3 page 213).

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 10.6 page 168).

13.16.1 <module> Exemple

Exemple utilisant la configuration par défaut du service web de SafeKit (protocole : HTTP, port : 9010) :

```
<check>
  <module name="mirror">
    <to  addr="M1host" port="9010"/>
  </module>
</check>
```

Exemple utilisant la configuration sécurisée du service web de SafeKit (protocole : HTTPS, port : 9453) :

```
<check>
  <module name="mirror">
    <to  addr="M1host" port="9453" secure="on"/>
  </module>
</check>
```

Important

Insérer le tag <module> dans la section <check> si celle-ci est déjà définie.

Pour d'autres exemples, voir les sections 15.3 page 279 et 15.13 page 291.

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. Par défaut external_module_name_<IP_@ or name of the server >.
[<to	Définition du/des serveurs exécutant le module externe. Par défaut, le serveur local.
addr="IP_@ or name of the server"	Adresse IP ou nom du 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> tag)

SafeKit offre un checker de splitbrain à utiliser pour des architectures miroir. Le splitbrain est une situation où, à la suite d'une panne matérielle ou logicielle, les deux nœuds SafeKit deviennent primaires, chaque nœud croyant 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 splitbrain, 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.

Important

Le ping entre les 2 nœuds et avec le witness doit être ouvert.

13.17.1 <splitbrain> Exemple

```
<check>
  <splitbrain ident="SBtest" exec="ping" arg="192.168.1.100" />
</check>
```

Important

Insérer le tag <splitbrain> dans la section <check> si celle-ci est déjà définie.

13.17.2 <splitbrain> Syntaxe

```
  <splitbrain
     ident="witness"
     exec="ping"
     arg="witness IP address "
  />
```

13.17.3 <splitbrain> Attributs

`<splitbrain`	Une seule section `<splitbrain>`.
`ident="witness"`	Nom de la ressource affichée par la commande `safekit state -v -m AM`. `splitbrain.witness` représente l'état du witness.
`[when="pre"]`	Valeur fixe. Le checker est démarré/arrêté après/avant les scripts `prestart`/`poststop`. Sur détection de splitbrain, le serveur pour lequel `splitbrain.witness="up"` devient primaire. Sur l'autre, pour lequel `splitbrain.witness="down"`, il y a affectation de la ressource `splitbrain.uptodate` à `"down"`. La règle de failover par défaut réalise un stopwait qui met le module dans l'état `WAIT` (pour plus d'information, voir 13.18 page 266). Chaque stopwait incrémente le compteur `maxloop` (voir 13.2.3 page 213).

exec="ping"	Valeur fixe. Utilise ping pour tester l'accessibilité au witness et affecte la ressource splitbrain.witness.
arg="IP_@ or name"	Adresse IP ou nom du witness Adresse IPv4 ou IPv6.
</splitbrain>	

13.18 Failover machine (<failover> tag)

SafeKit apporte des checkers (interface réseau, ping, tcp, custom, module). Un checker vérifie l'état d'une ressource (par défaut toutes les 10 secondes) et positionne son état à up ou down (voir section 13.10 page 255). La machine de failover 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 écrites dans un langage simple.

Dans un module ferme, la machine de failover ne fonctionne que sur les états locaux des ressources alors que dans un module miroir, elle peut fonctionner sur les états locaux et les états distants des ressources. Comme les états des ressources transitent par les voies de heartbeats, il est conseillé d'avoir plusieurs voies de heartbeats (voir section 13.3 page 215).

13.18.1 <failover> Exemple

```
<failover>
   <![CDATA[
     ping_failure: if (ping.testR2 == down) then stopstart();
   ]]>
</failover>
```

Voir aussi les règles de failover par défaut décrites en 13.18.5 page 267.

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"]	Avec extends="yes", les nouvelles règles de failover étendent les règles par défaut (voir 13.18.5 page 267). Avec 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> Commandes

`safekit set [-m AM] -r resource_class.resource_id -v resource_state`
 	
`[-n] [-l]`
	L'état d'une ressource est positionné par un checker avec cette commande
	Exemples :
	`safekit set -r custom.myresource -v up`
	`safekit set -r custom.myresource -v down`
	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

`safekit stopwait -i "identity"`	Equivalent à la commande `wait()` de la failover machine (voir 13.18 page 266). Avec stopwait, (1) `poststop` et `prestart` scripts ne sont pas appelés et (2) les checkers de type `when="pre"` ne sont pas stoppés.

Les autres commandes `restart()`, `stopstart()`, `stop()`, `swap()` sont équivalentes aux commandes de contrôle (avec le paramètre `-i identity` en plus) décrites en 9.4 page 148.

`maxloop` / `loop_interval` / `automatic_reboot` s'appliquent si le paramètre `-i identity` est passé aux commandes (voir 13.2 page 212). Ce qui est le cas lorsque les commandes sont appelées à partir de la failover machine.

13.18.5 Règles de failover

Les règles de failover par défaut pour les checkers 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 ping checkers */
ping_failure: if (ping.? == down) then wait();
```

```
/* rule for tcp checkers */
tcp_failure: if (tcp.? == down) then restart();
/* rule for ip checkers */
ip_failure: if (ip.? == down) then stopstart();
/* rules for splitbrain */
splitbrain_failure: if (splitbrain.uptodate == down) then wait();
]]>
</failover>
```

Elles sont définies dans le fichier `SAFE/private/conf/include/failover.xml`. Il existe en plus des règles de failover dédiées à la gestion de la réplication.

La commande `WAKEUP` est automatiquement générée lorsqu'aucune action `wait()` n'est applicable.

Depuis SafeKit 7.5, les règles de failover utilisent une nouvelle syntaxe et les règles pour la réplication sont définies dans un autre fichier `SAFE/private/conf/include/rfs.xml`.

En complément des règles par défaut, l'utilisateur peut définir ses propres règles de (pour un custom checker par exemple) en respectant la syntaxe suivante :

```
label: if ( expression ) then action;
```

with:

- `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`

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 | module | heartbeat | rfs
resource_id ::= * | ? | name
resource_state ::= init | down | up | unknown
```

`init`	Etat 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`	Etat OK
`down`	Etat KO
`unknown`	Etat inconnu pour une ressource distante (module arrêté sur l'autre nœud)

14. Scripts du module pour la configuration du module

Pour activer l'appel des scripts du module, le tag `<user>` doit être présent dans `userconfig.xml` (voir 13.7 page 245).

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).

Des exemples de scripts sont donnés en 15.1 page 276 pour un module miroir, et en 15.2 page 277 pour un module ferme.

Note

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 Note: Quand le serveur SECOND devient primaire, stop_second suivi de start_prim est exécuté
start_sec stop_sec	**Scripts spéciaux pour un module miroir.** Voir l'automate d'exécution des scripts décrit en 14.2 page 271
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 14.2 page 271

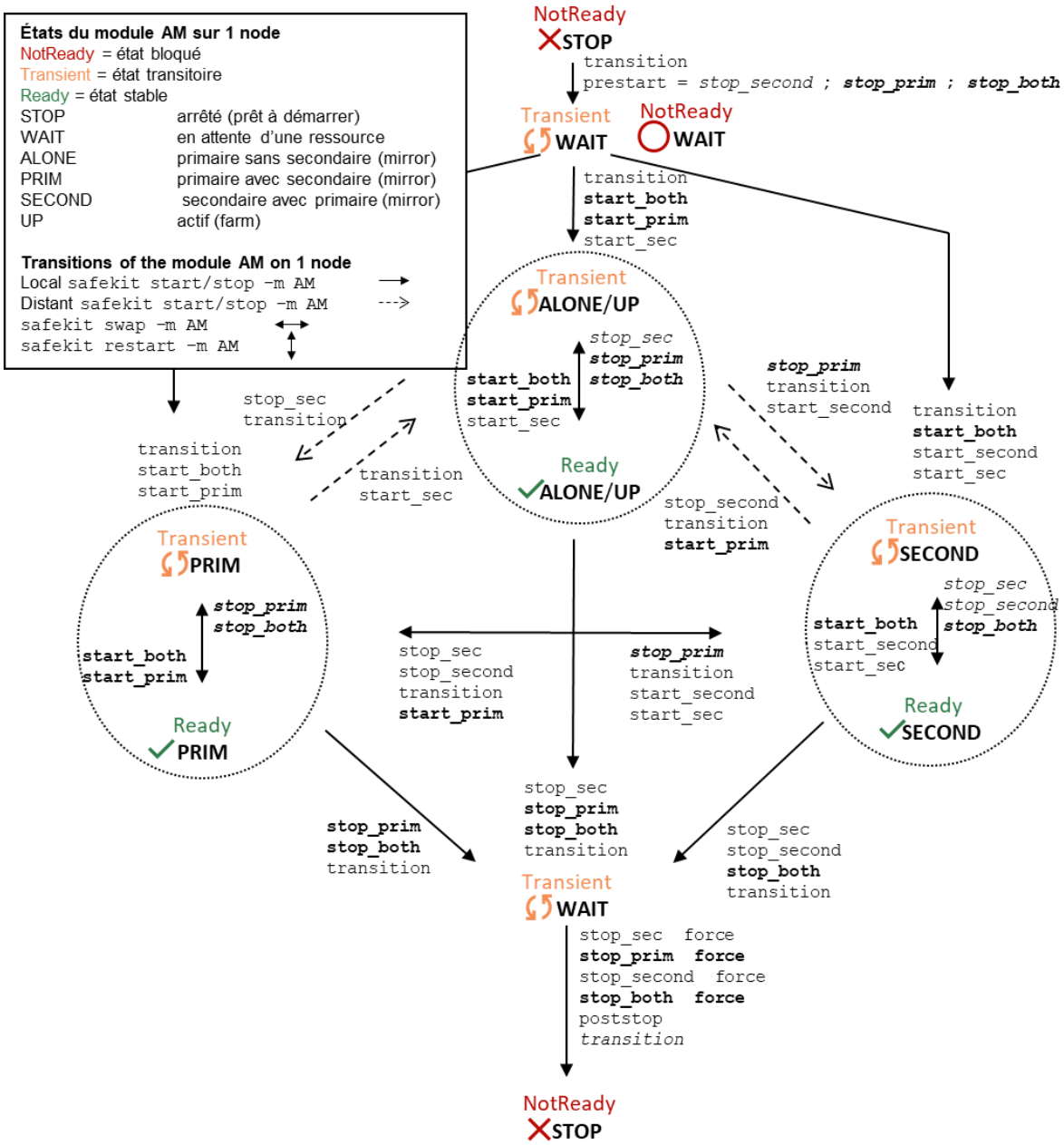
14.1.2 Autres scripts

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 Automate d'exécution des scripts

Exemple : au démarrage, la première transition de `STOP` vers `WAIT` appelle le script `transition STOP WAIT`.

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.



14.3 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`)

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`, `SAFEBIN`, `SAFEUSERBIN`, `SAFEUSERVAR` (voir 10.1 page 157)
- toutes les variables définies dans le tag <user> de `userconfig.xml` (voir 13.7 page 245).

14.4 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`. Note: A utiliser lorsque les commandes spéciales sont passées hors script SafeKit

14.4.1 Commandes pour Windows

14.4.1.1 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

14.4.1.2 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.

14.4.2 Commandes pour Linux

14.4.2.1 Gestion de la crontab

`$SAFEBIN/gencron`	
`[del | add]`	del pour désactiver des entrées dans `stop_prim` (en insérant des commentaires) ou add pour activer des entrées dans `start_prim` (en retirant les commentaires).
`<user name>`	Nom de l'utilisateur dans la crontab.
`[all |<command name>]`	S'applique à toutes les entrées ou au nom de la commande
`-c "<comment>"`	Entête du commentaire qui sera insérée.

Par exemple, pour désactiver/activer l'entrée de la crontab :

```
5 0 * * * $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
```

Insérer dans `stop_prim` :

```
$SAFEBIN/gencron del admin daily.job -c "SafeKit configuration for
$SAFEMODULE"
```

Et insérer dans `start_prim` :

```
$SAFEBIN/gencron add admin daily.job -c "SafeKit configuration for
$SAFEMODULE"
```

14.4.2.2 Commande bounding

```
$SAFEBIN/boundcmd <timeout value> <command path> [<args>]
```

Limite le temps d'exécution d'une 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
```

14.4.3 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> Liste (et arrête) les processus dont le nom de commande est celui spécifié et dont les arguments correspondent à l'expression régulière. safekit -r processtree kill notepad.exe ".*myfile.*" safekit -r processtree list all "mirror"
safekit incloop -m AM -i <handler name>	SafeKit fournit un compteur maxloop, du nombre de restart et stopstart 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, utilisez 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.
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 `userconfig.xml` et scripts du module

Certains exemples décrits sont tirés des modules livrés avec le package SafeKit, sous `SAFE/Application_Modules`. Vous pouvez les installer avec la console web (voir 3.3.1 page 45) pour examiner en détail leur contenu.

D'autres exemples d'intégration sont décrits sous https://www.evidian.com/fr/produits/haute-disponibilite-logiciel-clustering-application/configuration-cluster-basculement/.

Important

Les .safe sont plate-forme dépendants et, par conséquent, différents en Windows et Linux.

Tous les exemples utilisent la configuration du cluster SafeKit suivante (voir 12 page 205) :

```
<cluster>
     <lans name="net3">
         <node name="node1" addr="10.1.0.2"/>
         <node name="node2" addr="10.1.0.3"/>
         <node name="node3" addr="10.1.0.3"/>
      </lan>

      <lan name="default">
         <node name="node1" addr="192.168.1.1"/>
         <node name="node2" addr="192.168.1.2"/>
      </lan>
```

```
      <lan name="repli">
         <node name="node1" addr="10.0.0.2"/>
         <node name="node2" addr="10.0.0.3"/>
      </lan>
   </lans>
</cluster>
```

15.1 Exemple du module générique miroir avec `mirror.safe`

Ci-dessous le fichier de configuration et les scripts du module du module miroir générique, `mirror.safe`, pour Windows. Pour Linux, se référer au module `mirror.safe` livré avec le package Linux.

`conf/userconfig.xml` – voir 13 page 211

```
<!-- Mirror Architecture with Real Time File Replication and Failover -->
<!DOCTYPE safe>
<safe>
   <service mode="mirror" defaultprim="alone" maxloop="3" loop_interval="24"
failover="on">
      <heart pulse="700" timeout="30000">
         <heartbeat name="default" ident="flow"/>
      </heart>
      <rfs async="second" acl="off" locktimeout="200" nbrei="3" iotimeout="300">
         <replicated dir="c:\test1replicated" mode="read_only"/>
         <replicated dir="c:\test2replicated" mode="read_only"/>
      </rfs>
      <vip>
         <interface_list>
            <interface check="on" arpreroute="on">
               <real_interface>
                  <virtual_addr addr="192.168.4.10" where="one_side_alias"/>
               </real_interface>
            </interface>
         </interface_list>
      </vip>
      <user nicestoptimeout="300" forcestoptimeout="300" logging="userlog"/>
   </service>
</safe>
```

`bin/start_prim.cmd` – voir 14 page 269

```
@echo off

rem Script called on the primary server for starting application services

rem For logging into SafeKit log use:
rem "%SAFE%\safekit" printi | printe "message"

rem stdout goes into Application log
echo "Running start_prim %*"

set res=0

rem Fill with your services start call
rem net start "myservice" /Y
```

```
set res=%errorlevel%

if %res% == 0 goto end

:stop
"%SAFE%\safekit" printe "start_prim failed"

rem uncomment to stop SafeKit when critical
rem "%SAFE%\safekit" stop -i "start_prim"

:end
```

bin/stop_prim.cmd - voir 14 page 269

```
@echo off
rem Script called on the primary server for stopping application services

rem For logging into SafeKit log use:
rem "%SAFE%\safekit" printi | printe "message"

rem ----------------------------------------------------------
rem
rem 2 stop modes:
rem
rem - graceful stop
rem   call standard application stop with net stop
rem
rem - force stop (%1=force)
rem   kill application's processes
rem
rem ----------------------------------------------------------

rem stdout goes into Application log
echo "Running stop_prim %*"

set res=0

rem default: no action on forcestop
if "%1" == "force" goto end

rem Fill with your service(s) stop call
rem net stop "myservice" /Y

rem If necessary, uncomment to 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 du module générique 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 Linux, se référer au module `farm.safe` livré avec le package Linux.

conf/userconfig.xml - voir 13 page 211

```
<!-- Farm Architecture with Load-Balancing and Failover -->
<!DOCTYPE safe>
<safe>
   <service mode="farm" maxloop="3" loop_interval="24">
      <farm>
         <lan name=”default” />
      </farm>
      <vip>
         <interface_list>
            <interface check="on" arpreroute="on">
               <virtual_interface type="vmac_directed">
                  <virtual_addr addr="192.168.4.20" where="alias"/>
               </virtual_interface>
            </interface>
         </interface_list>
         <loadbalancing_list>
            <group name="FarmProto">
               <rule port="9010" proto="tcp" filter="on_port"/>
            </group>
         </loadbalancing_list>
      </vip>
      <user nicestoptimeout="300" forcestoptimeout="300" logging="userlog"/>
   </service>
</safe>
```

bin/start_both.cmd - voir 14 page 269

```
@echo off

rem Script called on all servers for starting applications

rem For logging into SafeKit log use:
rem "%SAFE%\safekit" printi | printe "message"

rem stdout goes into Application log
echo "Running start_both %*"

set res=0

rem Fill with your services start call
rem net start "myservice" /Y

set res=%errorlevel%

if %res% == 0 goto end

:stop
set res=%errorlevel%
"%SAFE%\safekit" printe "start_both failed"

rem uncomment to stop SafeKit when critical
rem "%SAFE%\safekit" stop -i "start_both"
:end
```

bin/stop_both.cmd - voir 14 page 269

```
@echo off
```

```
rem Script called on all servers for stopping application

rem For logging into SafeKit log use:
rem "%SAFE%\safekit" printi | printe "message"

rem -------------------------------------------------------
rem
rem 2 stop modes:
rem
rem - graceful stop
rem   call standard application stop with net stop
rem
rem - force stop (%1=force)
rem   kill application's processes
rem
rem -------------------------------------------------------

rem stdout goes into Application log
echo "Running stop_both %*"

set res=0

rem default: no action on forcestop
if "%1" == "force" goto end

rem Fill with your services stop call
rem net stop "myservice" /Y

rem If necessary, uncomment to wait for the stop of the services
rem "%SAFEBIN%\sleep" 10

if %res% == 0 goto end

"%SAFE%\safekit" printe "stop_both failed"
:end
```

15.3 Un module ferme dépendant d'un module miroir

Dans l'exemple ci-dessous, le module ferme ne peut démarrer qu'à condition que le module miroir soit opérationnel. Cette architecture peut être utilisée pour lier un module ferme IIS à un module miroir Microsoft SQL server. Elle est basée sur la configuration d'un module checker dans le module ferme. Pour plus détails, voir 13.16 page 263.

farm/conf/userconfig.xml - voir 13 page 211

```
…
  <!-- Checker Configuration: module dependency to mirror + local TCP checker -->
  <check>
    <module name="mirror">
     <to addr="192.168.1.31" port="9010/>
    </module>
   </check>
…
```

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.

15.4 Exemple d'un flux de réplication dédié

L'attribut `ident="flow"` sur le heartbeat, permet d'identifier le flux de réplication. Pour plus d'information, voir 13.6 page 226.

`conf/userconfig.xml` – voir 13 page 211

```
…
  <heart>
    <heartbeat name="default" />
    <!— 2nd heartbeat special for dedicated replicated network -->
    <heartbeat name="repli" ident="flow" />
  </heart>
…
```

15.5 Exemples de partage de charge dans un module ferme

15.5.1 Exemple d'un load balancing TCP

Avec le fichier de configuration **`userconfig.xml`** suivant, vous définissez une ferme de 3 serveurs avec partage de charge et reprise sur les ports TCP 9010 (service web SafeKit), 23 (Telnet), 80 (HTTP), 443 (HTTPS), 8080 (HTTP proxy) and 389 (LDAP).

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 1.4 page 19)

`conf/userconfig.xml` - voir 13 page 211

```
<!DOCTYPE safe>
<safe>
<service mode="farm">
  <farm>
    <lan name="net3" />
  </farm>
  <vip>
    <interface_list>
      <interface check="on" arpreroute="on">
        <virtual_interface type="vmac_directed">
          <virtual_addr addr="192.168.1.50" where="alias" />
        </virtual_interface>
      </interface>
    </interface_list>
    <loadbalancing_list>
       <group name="tcpservices" >
         <cluster>
           <host name="node1" power="1" />
           <host name="node2" power="1" />
```

```
          <host name="node3" power="1" />
        </cluster>
        <rule port="9010" proto="tcp" filter="on_port" />
        <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>
</service>
</safe>
```

15.5.2 Exemple de load balancing UDP

Avec le fichier **`userconfig.xml`** suivant, vous définissez une ferme de 3 serveurs avec partage de charge et reprise sur les services UDP 53 (DNS) et 1645 (RADIUS).

`conf/userconfig.xml` – voir 13 page 211

```
<!DOCTYPE safe>
<safe>
<service mode="farm">
<farm>
    <lan name="net3" />
  </farm>
  <vip>
    <interface_list>
      <interface check="on">
        <virtual_interface type="vmac_invisible">
          <virtual_addr addr="192.168.1.50" where="alias" />
        </virtual_interface>
      </interface>
    </interface_list>
    <loadbalancing_list>
      <group name="udpservices" >
        <cluster>
          <host name="node1" power="1" />
          <host name="node2" power="1" />
          <host name="node3" power="1" />
        </cluster>
        <rule port="53"    proto="udp" filter="on_ipid" />
        <rule port="1645"  proto="udp" filter="on_ipid" />
      </group>
    </loadbalancing_list>
  </vip>
</service>
</safe>
```

Note

Avec `on_ipid`, le load balacing est réalisé sur le champ "IP identifier" dans l'entête IP du paquet. Le load balancing fonctionne même si le client présente la même adresse IP client et le même port en entrée.

15.5.3 Exemple d'un load balancing multi-groupes

Avec le fichier **userconfig.xml** suivant, vous définissez une ferme de 3 serveurs avec une priorité pour le trafic HTTP pour node1, HTTPS pour node2 et proxy HTTP pour le node3.

conf/userconfig.xml - voir 13 page 211

```
<!DOCTYPE safe>
<safe>
<service mode="farm">
  <farm>
    <lan name="net3" />
  </farm>
  <vip>
    <interface_list>
      <interface check="on">
        <virtual_interface type="vmac_invisible">
          <virtual_addr addr="192.168.1.50" where="alias" />
        </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>
</service>
</safe>
```

15.6 Exemple d'un hostname virtuel avec `vhost.safe`

Le module de démonstration `vhost.safe` montre comment positionner un hostname virtuel (voir 13.8 page 246).

`conf/userconfig.xml` – voir 13 page 211

```
…
  <vhost>
    <virtualhostname name="virtualname" envfile="vhostenv.cmd" />
  </vhost>
…
```

En plus de cette configuration, il faut exécuter des commandes spéciales dans les scripts du module. Ci-dessous l'exemple des scripts Windows. Pour Linux, se référer au `vhost.safe` livré avec la package Linux.

`bin/start_prim.cmd` - voir 14 page 269

```
@echo off

rem Script called on the primary server for starting application services

rem For logging into SafeKit log use:
rem "%SAFE%\safekit" printi | printe "message"

rem stdout goes into Application log
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 Fill with your services start call

set res=%errorlevel%

if %res% == 0 goto end

:stop
"%SAFE%\safekit" printe "start_prim failed"

rem uncomment to stop SafeKit when critical
rem "%SAFE%\safekit" stop -i "start_prim"
:end
```

bin/stop_prim.cmd - voir 14 page 269

```
@echo off
rem Script called on the primary server for stopping application services

rem For logging into SafeKit log use:
rem "%SAFE%\safekit" printi | printe "message"

rem ----------------------------------------------------------
rem
rem 2 stop modes:
rem
rem - graceful stop
rem   call standard application stop with net stop
rem
rem - force stop (%1=force)
rem   kill application's processes
rem
rem ----------------------------------------------------------

rem stdout goes into Application log
echo "Running stop_prim %*"

set res=0

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%

rem default: no action on forcestop
if "%1" == "force" goto end

rem Fill with your services stop call

rem If necessary, uncomment to wait for the stop of the services
rem "%SAFEBIN%\sleep" 10


if %res% == 0 goto end

"%SAFE%\safekit" printi "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
```

15.7 Détection de la mort de processus avec softerrd.safe

Le module softerrd.safe est un module de démonstration de la surveillance de la mort de processus (pour plus d'information, voir 13.9 page 248).

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

La détection de l'arrêt de :

- mybin provoque un restart du module
- myappli provoque l'exécution d'un handler spécial restart_myappli.cmd. Ce script incrémente le compteur maxloop et redémarre le processus myappli
- myotherbin provoque un stop du module

Les tests consistent à tuer les processus mybin, myotherbin ou myappli avec la commande safekit kill.

Ci-dessous un extrait de softerrd.safe pour Windows. Pour Linux, regardez celui livré avec le package Linux.

conf/userconfig.xml - voir 13 page 211

```
…
      <errd>
         <proc name="mybin.exe" atleast="1" action="restart" class="prim"/>
         <proc name="myotherbin.exe" atleast="1" action="stop" class="second"/>
         <proc name="myappli.exe" atleast="1" action="restart_myappli"
class="myappli"/>
      </errd>
…
```

bin/start_prim.cmd - voir 14 page 269

Noter l'appel à %SAFE%\safekit errd enable myappli pour commencer la surveillance des processus avec class="myappli"

```
@echo off

%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

:end
```

bin/stop_prim.cmd - voir 14 page 269

Noter l'appel à `%SAFE%\safekit errd disable myappli` pour arrêter la surveillance des processus avec `class="myappli"`

```
@echo off
rem default: no action on forcestop
if "%1" == "force" goto end

%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
```

bin/restart_myappli.cmd

Noter l'incrémentation du compteur de rebouclage et l'arrêt du module quand `maxloop` est atteint

```
@echo off

rem Template for script called by errd on error detection instead of standard
restart
%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
```

15.8 Exemple d'un checker TCP

Le checker tcp teste le service web Apache (pour plus d'information, voir 13.11 page 256). L'action par défaut quand le service TCP est down est de redémarrer le module (voir 13.18.5 page 267).

conf/userconfig.xml - voir 13 page 211

```
…
  <check>
    <tcp
          ident="Apache_80"
          when="both"
    >
          <to
          addr="172.21.10.5"
          port="80"
          interval="120"
          timeout="5"
          />
    </tcp>
   </check>
…
```

15.9 Exemple d'un checker ping

Le checker ping suivant teste un routeur d'adresse IP 192.168.1.1 (pour plus d'information, voir 13.12 page 257). L'action par défaut lorsque le routeur est down et de stopper localement le module et d'attendre que le ping redevienne up (voir 13.18.5 page 267).

conf/userconfig.xml - voir 13 page 211

```
…
<check >
  <ping ident="router">
  <to addr="192.168.1.1"/>
  </ping>
</check>
…
```

15.10 Exemple d'un 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 13.5 page 219). Dans le fichier de configuration, l'adresse virtuelle est définie comme suit :

conf/userconfig.xml - voir 13 page 211

```
…
<vip>
 <interface_list>
  <interface check="on">
   <real_interface>
    <virtual_addr addr="192.168.1.32" where="one_side_alias"/>
   </real_interface>
  </interface>
 </interface_list>
```

```
</vip>
…
```

L'action par défaut lorsque l'interface est down et de stopper localement le module et d'attendre que l'interface redevienne up (voir 13.18.5 page 267).

➪ Configuration générée en Windows

```
<check>
 <intf when="pre" ident="192.168.1.0"
       intf="{8358A0EE-2F3F-4FEE-A33B-EDC406C0C858}">
  <to local_addr="192.168.1.228"/>
 </intf>
</check>
```

Où `{8358A0EE-2F3F-4FEE-A33B-EDC406C0C858}` est l'identité de l'interface réseau avec le réseau 192.168.1.0 et avec 192.168.228 comme première adresse IP (`safekit -r vip_if_ctrl -L`).

➪ Configuration générée en Linux

```
<check>
 <intf when="pre" ident="192.168.1.0" intf="eth2">
  <to local_addr="192.168.1.20"/>
 </intf>
</check>
```

Où eth2 est l'identité de l'interface réseau avec le réseau 192.168.1.0 et avec 192.168.228 comme première adresse IP (`ifconfig -a` ou `ip addr show`).

Pour plus de détails, voir 13.13 page 258.

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 13.5 page 219). Dans le fichier `userconfig.xml`, l'adresse virtuelle est définie comme suit :

`conf/userconfig.xml` - voir 13 page 211

```
…
<vip>
 <interface_list>
  <interface check="on" arpreroute="on">
   <real_interface>
    <virtual_addr addr="192.168.1.99" where="one_side_alias" check="on"/>
   </real_interface>
  </interface>
 </interface_list>
</vip>
…
```

L'action par défaut lorsque le checker détecte que l'adresse n'est plus configurée est d'exécuter un stopstart du module (voir 13.18.5 page 267).

- Configuration générée

```
<check>
<ip ident="192.168.1.99" when="prim">
<to addr="192.168.1.99"/>
</ip>
</check>
```

Pour plus de détails, voir 13.14 page 259.

15.12 Exemple d'un checker customisé avec `customchecker.safe`

Le module `customchecker.safe` est un module de démonstration d'un checker customisé dans un module miroir (pour plus d'information, voir section 13.15 page 261).

- Le checker teste la présence d'un fichier sur le serveur primaire (`when="prim"`). La ressource associée s'appelle `custom.checkfile` (`ident="checkfile"`). Elle est affectée à `up` quand le fichier est présent à down `sinon`.
- La règle de failover associée (configuré dans `<failover>`), nommée `c_checkfile`, provoque le `restart` du module si le fichier est absent (voir 13.18.5 page 267). A partir de SafeKit 8, la règle de failover associée est générée automatiquement en fonction de la valeur de l'attribut `action`.

Cet exemple peut servir de base pour l'écriture de votre propre checker.

`conf/userconfig.xml` pour SafeKit >= 8 - voir 13 page 211

```
…
  <check>
    <custom ident="checkfile" exec="checker.ps1"
          arg="c:\safekit\checkfile" when="prim" action="restart" />
  </check>
  <user></user>
…
```

`conf/userconfig.xml` pour SafeKit < 8 - voir 13 page 211

```
…
  <check>
    <custom ident="checkfile" exec="checker.ps1"
          arg="c:\safekit\checkfile" when="prim"/>
  </check>
  <user></user>
  <failover>
     <![CDATA[
       c_checkfile:
         if( custom.checkfile == down ) then restart();
         ]]>
  </failover>
…
```

`bin/checker.ps1`

Noter l'appel à `safekit set -r custom.checkfile -m AM` pour affecter l'état de la ressource (`up` or `down`)

```
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)
{
         $res="down"
    # Replace the following by your test
    if (Test-Path "$Arg1Value")
         {
         $res="up"
         }
         return $res
}

$customchecker=$MyInvocation.MyCommand.Name
$safekit="$env:SAFE/safekit.exe"
$safebin="$env:SAFEBIN"
$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
         }
}
```

L'exécutable associé au 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 script `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.

Important

A chaque fois que vous modifiez le fichier de configuration ou le script, vous devez réappliquer la nouvelle configuration.

15.13 Exemple d'un checker de module 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 13 page 211

```
…
<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>
…
```

Important Si vous avez modifié le port d'écoute du service web de SafeKit (voir 10.6 page 168), remplacez la configuration synthétique par la configuration complète et changez la valeur du port.

15.14 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 changements d'état du module principal. 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.

Important A chaque fois que vous modifiez un script du module, vous devez appliquer la nouvelle configuration.

15.14.1 Notification sur démarrage et arrêt du module

Les lignes suivantes, insérées à la fin du script de prestart du module (nommé AM), envoient un e-mail avec le nom du module et du serveur sur lequel le module est démarré.

➪ En Windows: **c:\safekit\modules\AM\bin\prestart.ps1**

```
$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'
}
```

➪ En Linux: **/opt/safekit/modules/AM/bin/prestart**

```
if [ "$3" = "start" ]; then
  echo "*** Start of module $SAFEMODULE on " `hostname`
  # insert here your notification: the module is starting
  #echo "Running prestart" | mail -s " Start of module $SAFEMODULE on
`hostname`" admin@mydomain.com
fi
```

Lorsque le module s'arrête, il peut être notifié à l'aide du script poststop. Celui-ci n'est pas livré par défaut et peut être créé comme suit.

➪ En Windows: **c:\safekit\modules\AM\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
}
```

➪ En Linux: **/opt/safekit/modules/AM/bin/poststop**

```
#!/bin/sh
# Script called on module stop
# after resetting SafeKit resources

# For logging into SaKit log use:
# $SAFE/safekit printi | printe "message"

echo "Running poststop $*"

if [ "$3" = "stop" ]; then
  echo "*** Stop of module $SAFEMODULE on " `hostname`
  # insert here your notification: the module is stopping
  #echo "Running poststop" | mail -s " Stop of module $SAFEMODULE on `hostname`"
admin@mydomain.com
fi
```

15.14.2 Notification sur changement d'état du module

Le script `transition` peut être utilisée pour envoyer un e-mail 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 et peut être créé comme suit.

Pour un module ferme, changer les valeurs des états.

➪ En Windows: **c:\safekit\modules\AM\bin\transition.ps1**

```
# 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
}
```

➪ En Linux: **/opt/safekit/modules/AM/bin/transition**

```
#!/bin/sh
# Script called on module state change

# For logging into SaKit log use:
# $SAFE/safekit printi | printe "message"

echo "Running transition $*"

hostname=`hostname`

if [ "$1" = "WAIT" -a "$2" = "ALONE" ] ; then
```

```
  echo "*** Start ALONE of module $SAFEMODULE on $hostname"
  # insert here your notification: the module is starting as ALONE
  #echo "Running poststop" | mail -s " Start ALONE of module $SAFEMODULE on
$hostname" admin@mydomain.com
fi
if [ "$1" = "WAIT" -a "$2" = "PRIM" ] ; then
  echo "*** Start PRIM of module $SAFEMODULE on $hostname"
  # insert here your notification: the module is starting as PRIM
  #echo "Running poststop" | mail -s " Start PRIM of module $SAFEMODULE on
$hostname" admin@mydomain.com
fi
if [ "$1" = "WAIT" -a "$2" = "SECOND" ] ; then
  echo "*** Start SECOND of module $SAFEMODULE on $hostname"
  # insert here your notification: the module is starting as SECOND
  #echo "Running poststop" | mail -s " Start SECOND of module $SAFEMODULE on
$hostname" admin@mydomain.com
fi

if [ "$1" != "WAIT" -a "$2" = "ALONE" ] ; then
  echo "*** Go ALONE of module $SAFEMODULE on $hostname"
  # insert here your notification: the module is going ALONE
  #echo "Running poststop" | mail -s " Go ALONE of module $SAFEMODULE on
$hostname" admin@mydomain.com
fi
if [ "$1" != "WAIT" -a "$2" = "PRIM" ] ; then
  echo "*** Go PRIM of module $SAFEMODULE on $hostname"
  # insert here your notification: the module is going PRIM
  #echo "Running poststop" | mail -s " Go PRIM of module $SAFEMODULE on
$hostname" admin@mydomain.com
fi
if [ "$1" != "WAIT" -a "$2" = "SECOND" ] ; then
  echo "*** Go SECOND of module $SAFEMODULE on $hostname"
  # insert here your notification: the module is going SECOND
  #echo "Running poststop" | mail -s " Go SECOND of module $SAFEMODULE on
$hostname" admin@mydomain.com
fi
```

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 page 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 10.3.3.2 page 163
- 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 page 177)

Paramétrage de SafeKit

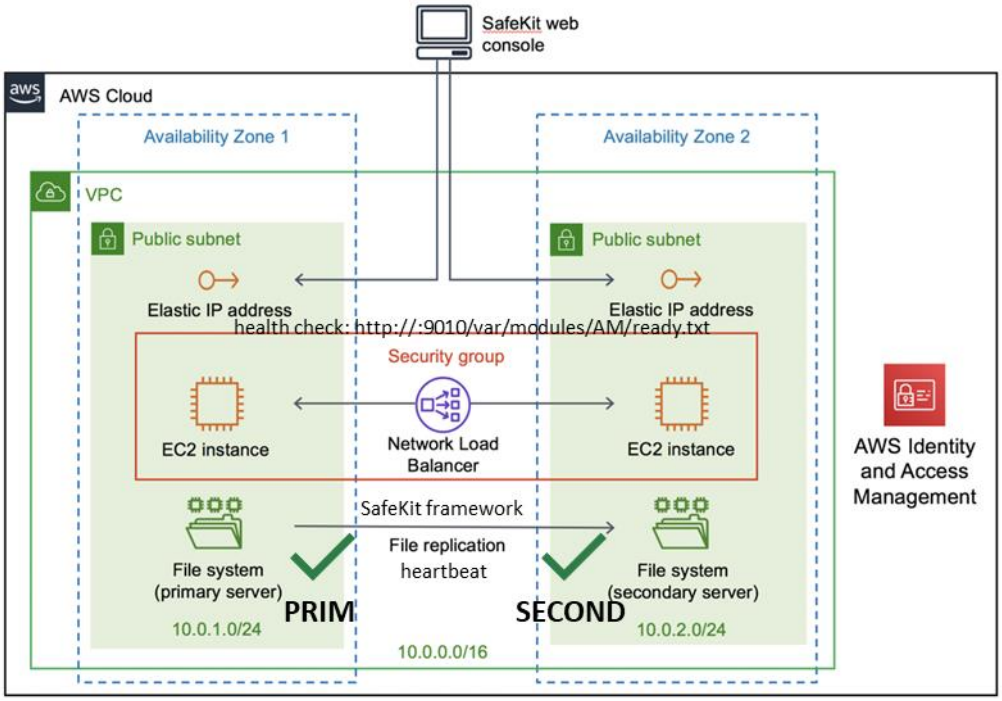
Enfin il faut saisir la configuration du cluster SafeKit et l'appliquer à tous les nœuds (voir 12 page 205). Par exemple, le fichier de configuration du cluster SafeKit serait :

```
<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

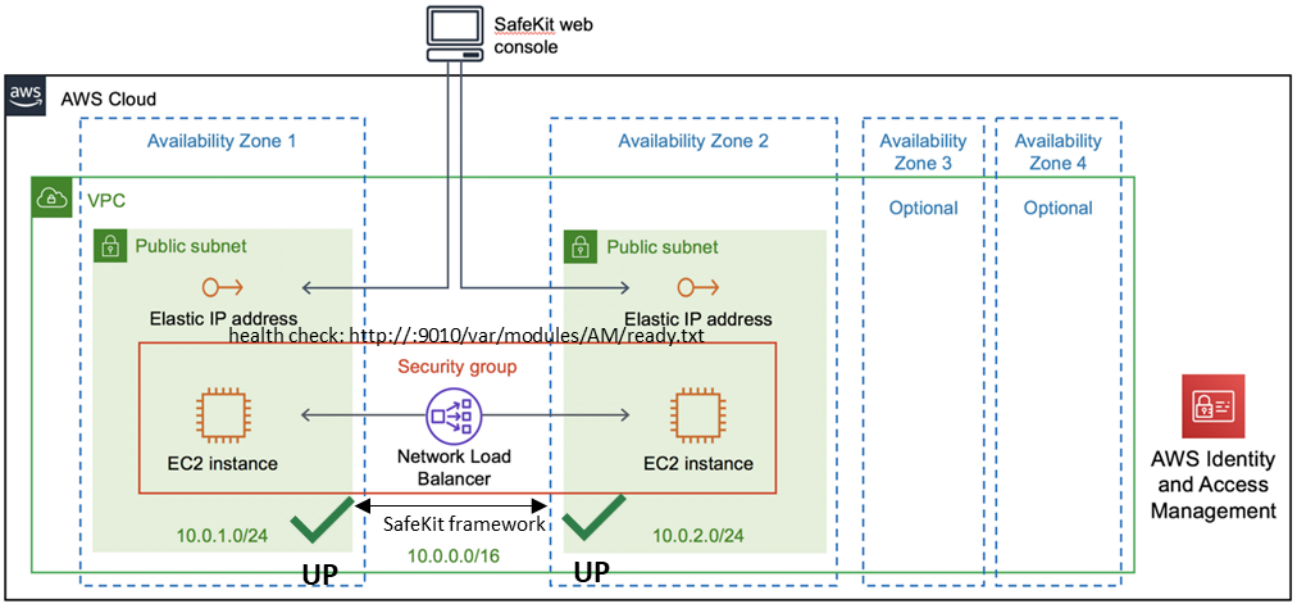
  TCP – 9453 pour la console web SafeKit en HTTPS
- TCP – 9001 pour configurer la console web SafeKit en HTTPS

Important

Les valeurs de ports du module dépendent de son id (voir 10.3.3.2 page 163). 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 10.3.3.2 page 163
- 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 page 177)

Paramétrage de SafeKit

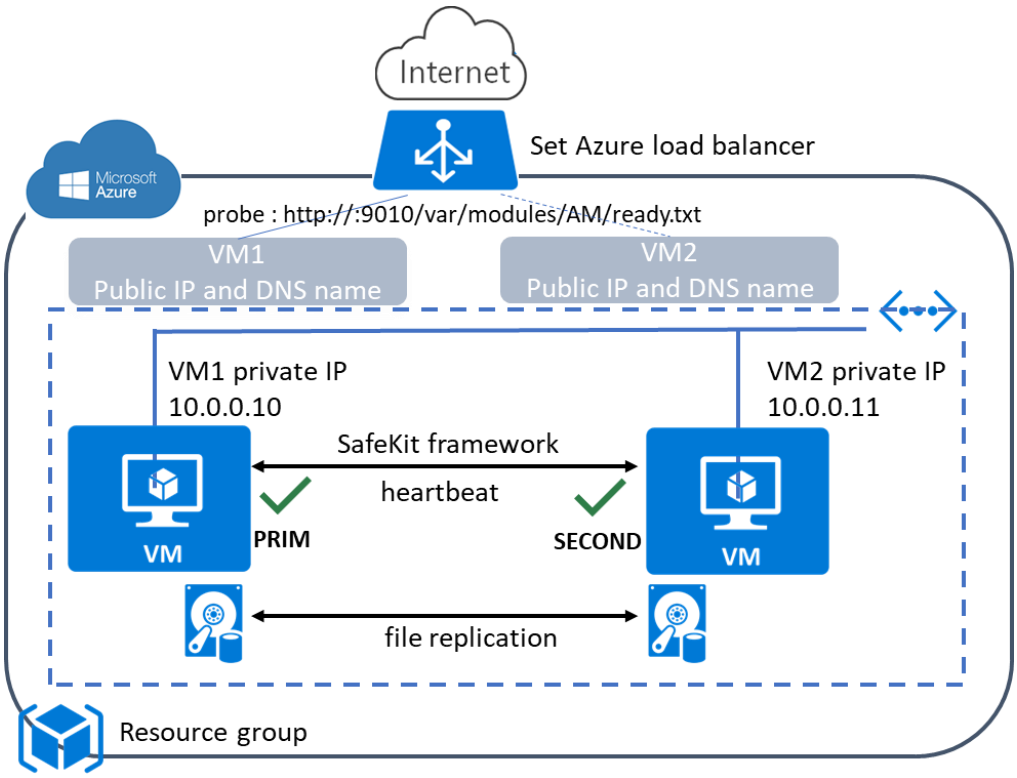
Enfin il faut saisir la configuration du cluster SafeKit et l'appliquer à tous les nœuds (voir 12 page 205). Par exemple, le fichier de configuration du cluster SafeKit serait :

```
<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

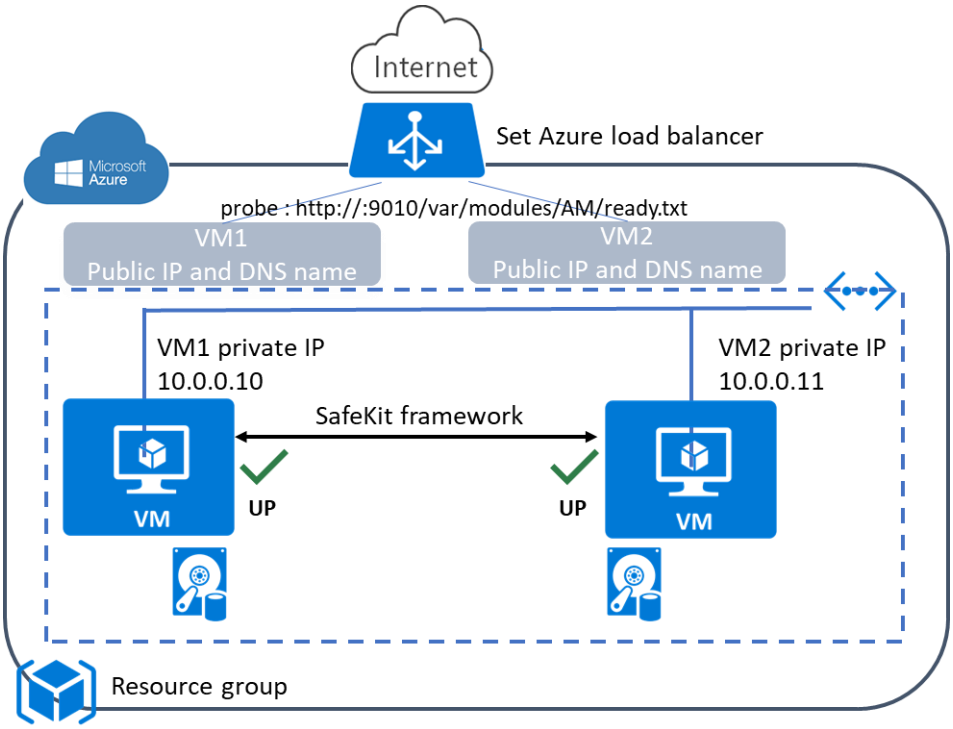
  TCP – 9453 pour la console web SafeKit en HTTPS
- TCP – 9001 pour configurer la console web SafeKit en HTTPS

Important

Les valeurs de ports du module dépendent de son id (voir 10.3.3.2 page 163). 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 du 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 10.3.3.2 page 163
- 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 page 177)

Paramétrage de SafeKit

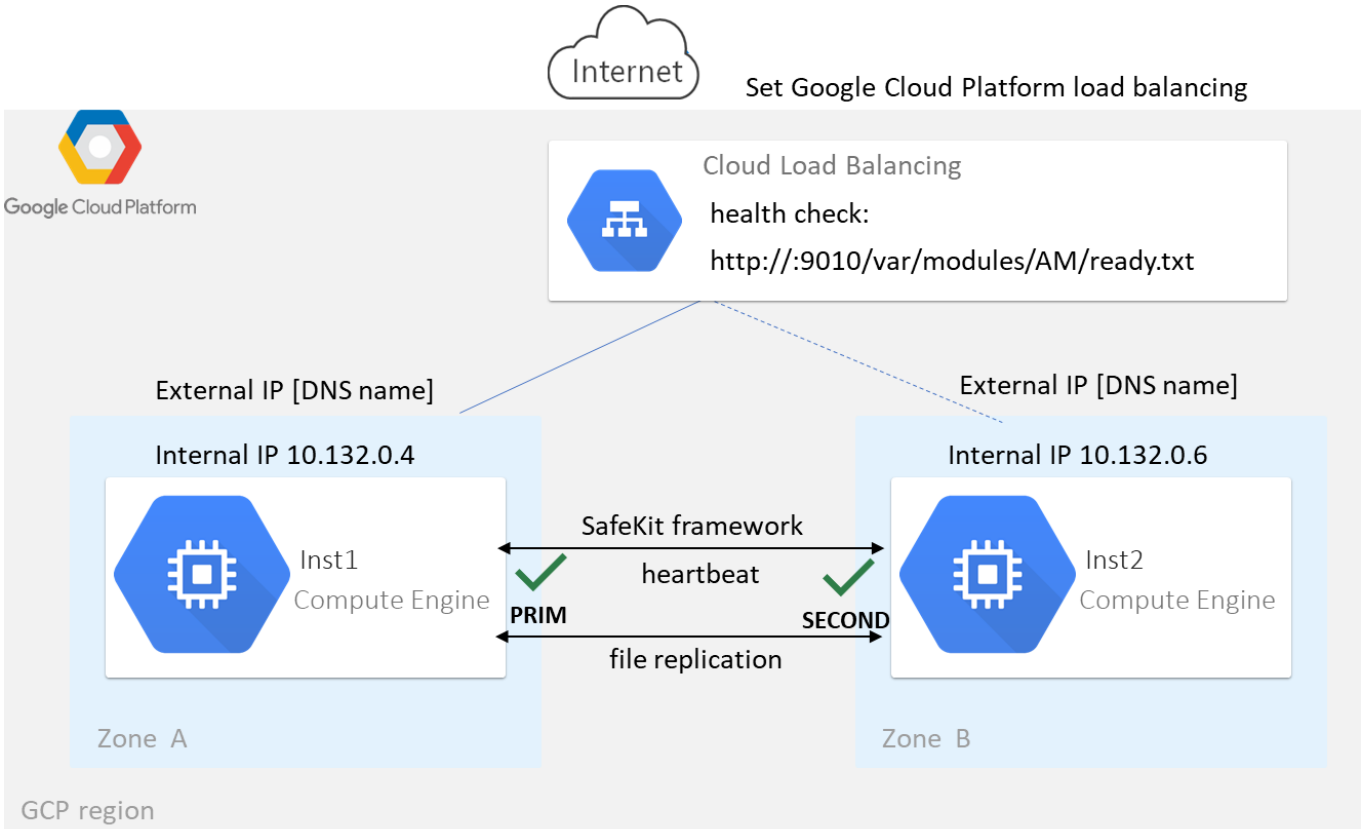
Enfin il faut saisir la configuration du cluster SafeKit et l'appliquer à tous les nœuds (voir 12 page 205). Par exemple, le fichier de configuration du cluster SafeKit serait :

```
<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

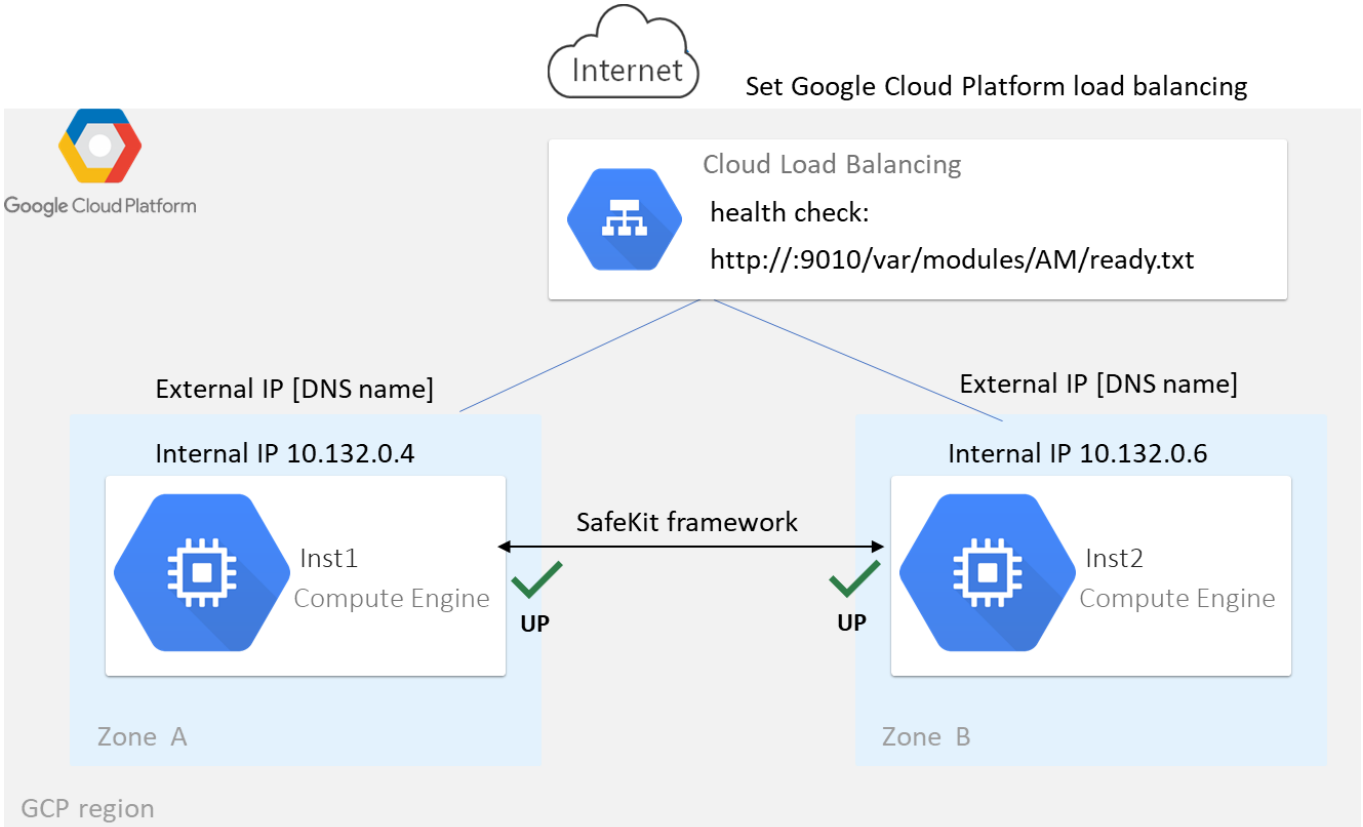
  TCP – 9453 pour la console web SafeKit en HTTPS
- TCP – 9001 pour configurer la console web SafeKit en HTTPS

Important

Les valeurs de ports du module dépendent de son id (voir 10.3.3.2 page 163). 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-page 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/htdcos/console/fr/3rdpartylicenses.txt .

Index des messages du journal du module

"Action ..."

Réplication et réintegration

Load-balancing

"Local state ..."

"Remote state ..."

"Resource ..."

"Script ..."

"Transition ..."

Autres messages

Index