NOTE TECHNIQUE

STORMSHIELD NETWORK SECURITY

ROUTAGE DYNAMIQUE BIRD V2

Produits concernés : SNS 4.8.1 et versions supérieures

Dernière mise à jour du document : 25 septembre 2024

Référence : sns-fr-routage_dynamique_BIRD_v2_note_technique

Table des matières

Historique des modifications 4
Introduction 5
Limitation 5
Comprendre le module Routage Dynamique 6
L'onglet Général 6
L'onglet BIRD v2 7
L'onglet BIRD v1 IPv4 8
L'onglet optionnel BIRD v1 IPv6 9
La console de contrôle 9
Connaître l'environnement BIRD / Stormshield Network 10
Démarrer le routage BIRD v2 depuis l'interface Web d'administration 10
Contrôler le routage dynamique BIRD v2 en mode interactif 10
Configurer le routage dynamique 14
Connaître les règles de syntaxe 15
Vérifier une configuration 15
Interagir avec le routage Stormshield Network 16
Comprendre des configurations simples 18
RIP 18
Autoriser le protocole RIP dans la politique de filtrage 19
Vérifier le bon fonctionnement du routage dynamique RIP 19
OSPF 20
Autoriser le protocole OSPF dans la politique de filtrage 21
Vérifier le bon fonctionnement du routage dynamique OSPF 22
BGP 23
Explications 24
Autoriser le protocole BGP dans la politique de filtrage 25
Vérifier le bon fonctionnement du routage dynamique BGP 25
Authentification 25
Cas d'une configuration en haute disponibilité (cluster) 26
Configuration avancée 27
Configuration BIRD 28
Autoriser les protocoles RIP, BGP et OSPF dans la politique de filtrage 30
Vérifier le bon fonctionnement du routage dynamique 30
Cas d'une configuration en haute disponibilité (cluster) 32
Migrer une configuration de routage dynamique depuis BIRD v1 vers BIRD v2 33
Comprendre le module Routage Dynamique 33
L'onglet Général 33
L'onglet BIRD v2 34
L'onglet BIRD v1 IPv4 35
L'onglet optionnel BIRD v1 IPv6 36
La console de contrôle 36
Réaliser la migration BIRD v1 vers BIRD v2 36
Préparer la configuration de BIRD v2 37
Contrôler le fonctionnement du routage dynamique 37

Annexe A : Tunnels VPN Hub and Spoke routés via BGP 38
Configuration des tunnels 38
Configuration BGP du site principal (Hub) 39
Configuration BGP du site satellite Spoke A 40
Configuration BGP du site satellite Spoke B 41
Vérification des tables de routage 41
Annexe B : Connectivité Amazon VPC 43
Configuration Amazon 43
Configuration des tunnels 44
Configuration BGP 44
Pour aller plus loin 46

Historique des modifications

Date	Description
25 septembre 2024	Nouveau document

Introduction

La version SNS 4.8.1 introduit le support du moteur de routage dynamique BIRD v2 destiné à remplacer BIRD v1 devenu obsolète.

Ce document a pour objet de guider l'administrateur d'un firewall SNS dans la configuration et l'exploitation du module de routage dynamique intégré BIRD v2.

Si vous souhaitez migrer votre configuration BIRD v1 vers BIRD v2, référez-vous à la section Migrer une configuration BIRD v1 vers BIRD v2

Il décrit dans un premier temps l'environnement de configuration ainsi que les modes d'interaction avec le moteur de routage. Il présente ensuite une configuration typique simple pour les trois protocoles de routage BGP, RIP et OSPF.
Ces exemples sont l'occasion de découvrir la structure de configuration des protocoles ainsi que des éléments périphériques, filtrage et affichage des états. La dernière section se concentre sur une configuration plus complexe.

Notez que BIRD propose de multiples options de configuration notamment pour l'échange de routes entre process, leur filtrage ou une pseudo virtualisation des instances de routage. Ces éléments avancés et spécifiques à BIRD sont exclus du périmètre du document.

Ces éléments ainsi que la syntaxe détaillée de la configuration du routage dynamique sont disponibles dans le manuel utilisateur BIRD v2 publié sur le site Web de l'éditeur de BIRD.

Limitation

IMPORTANT
Dans le cas où votre parc de firewalls SNS est géré par un serveur SMC, vous devez disposer d'une version de SMC 3.6 ou supérieure pour gérer le routage dynamique de vos firewalls en versions 4.8.1 et supérieures.

Comprendre le module Routage Dynamique

Vous pouvez configurer le routage dynamique dans l'onglet **Général** du module **Routage Dynamique** de l'interface Web d'administration.

Les onglets **BIRD v2**, **BIRD v1 IPv4** et **BIRD v1 IPv6** (si le support IPv6 est activé) permettent l'édition des fichiers de configuration de BIRD.

Notez que l'éditeur de l'interface graphique ne permet pas d'accéder aux modes interactifs permettant le contrôle du routage dynamique (tests de fonctionnement de nouvelle configuration via une configuration temporaire et visualisation des états).

Le tableau ci-dessous met en correspondance chaque version de BIRD, son onglet de configuration dans l'interface Web d'administration, son fichier de configuration et son binaire du mode interactif en mode console :

Version de BIRD	Onglet de configuration	Fichier de configuration	Binaire interactif
BIRD v2 - IPv4 et IPv6	BIRD v2	bird.conf	birdc
BIRD v1 - IPv4	BIRD v1 IPv4	bird4.conf	birdc4
BIRD v1 - IPv6	BIRD v1 IPv6	bird6.conf	birdc6

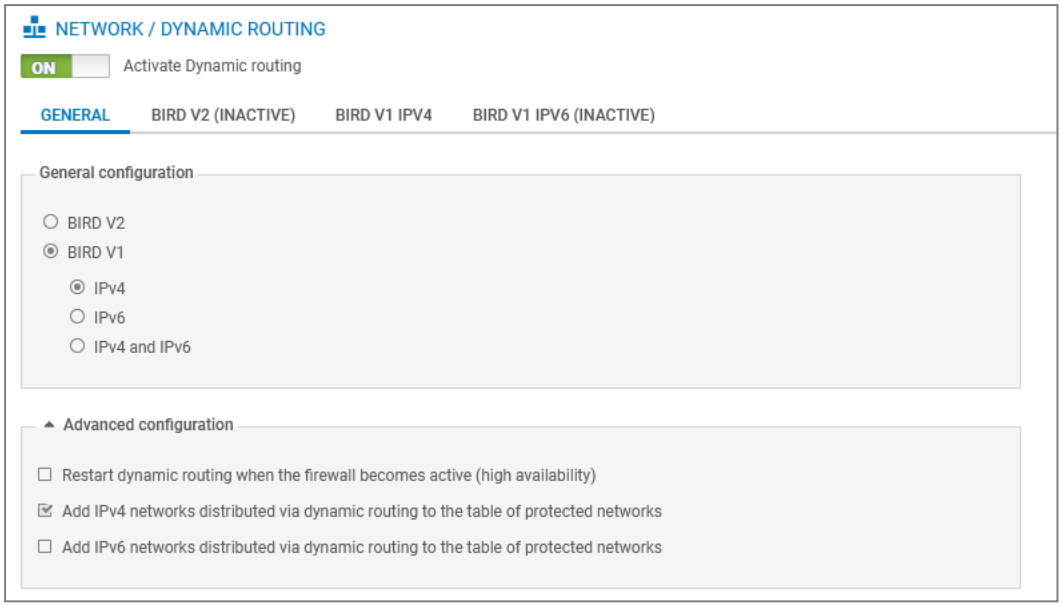
NOTE
Lorsqu'une version de BIRD est désactivée, l'onglet de configuration correspondant affiche la mention "(INACTIVE)".
Exemple : BIRD v2 (INACTIVE).

L'onglet Général

Cet onglet vous permet d'activer / désactiver la version souhaitée du moteur de routage dynamique BIRD.

Après la mise à jour d'un firewall SNS d'une version antérieure à SNS 4.8.1 vers une version SNS 4.8.1 ou supérieure, la configuration est la suivante :

- **BIRD v2** : ce bouton radio est sélectionné par défaut.
- **BIRD v1** : ce bouton radio est sélectionné si le firewall était initialement configuré uniquement en IPv4 s'il possédait une configuration BIRD v1 IPv4 active avant la mise à jour de firmware.
  Les boutons radio suivants ne sont affichés que si le firewall était initialement configuré en IPv4 et IPv6 :
  - **IPv4** : ce bouton radio est sélectionné pour un firewall sur lequel seule une configuration BIRD v1 IPv4 était active avant la mise à jour de firmware.
  - **IPv6** : ce bouton radio est sélectionné pour un firewall sur lequel seule une configuration BIRD v1 IPv6 était active avant la mise à jour de firmware.
  - **IPv4 et IPv6** : ce bouton radio est sélectionné pour un firewall sur lequel des configuration BIRD v1 IPv4 et IPv6 étaient actives avant la mise à jour de firmware.



IMPORTANT

Si vous souhaitez que les routes apprises par BIRD soient ajoutées automatiquement à la table des réseaux protégés et éviter ainsi de générer à tort des alertes d'*antispoofing* concernant ces réseaux, cochez ces cases (suivant votre configuration) :

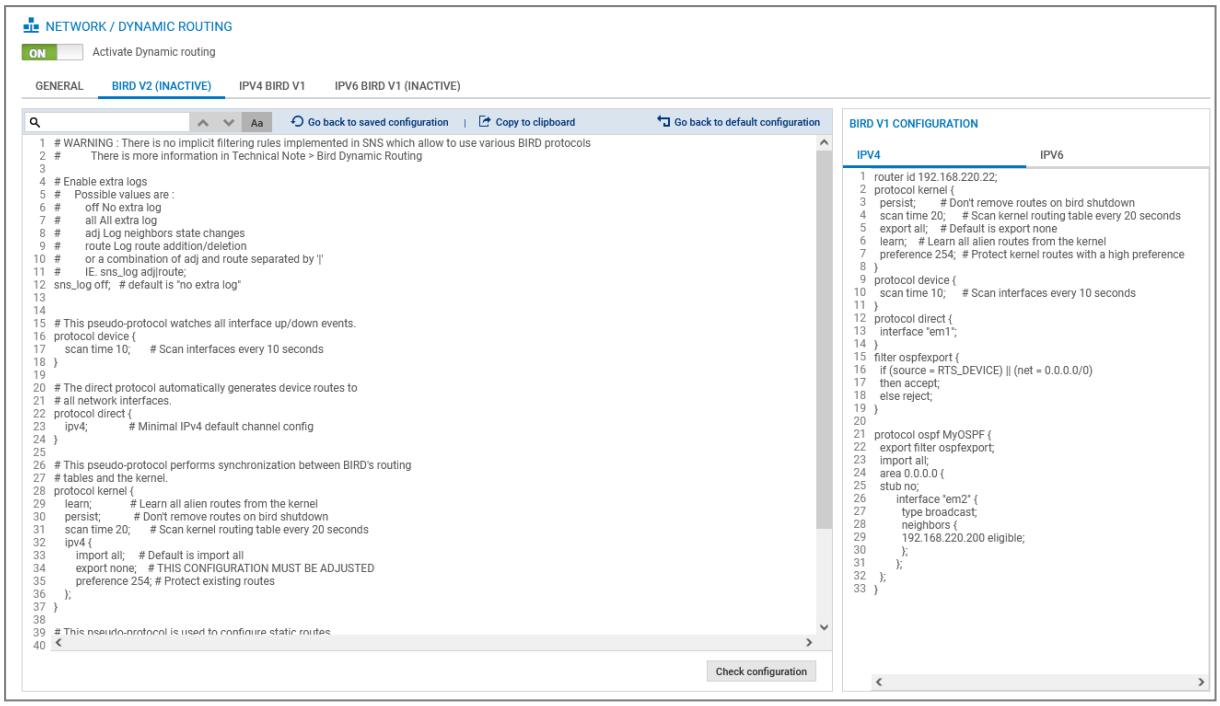
- **Ajouter les réseaux IPv4 distribués par le routage dynamique dans la table des réseaux protégés**.
- **Ajouter les réseaux IPv6 distribués par le routage dynamique dans la table des réseaux protégés**.

L'onglet BIRD v2

Cet onglet affiche :

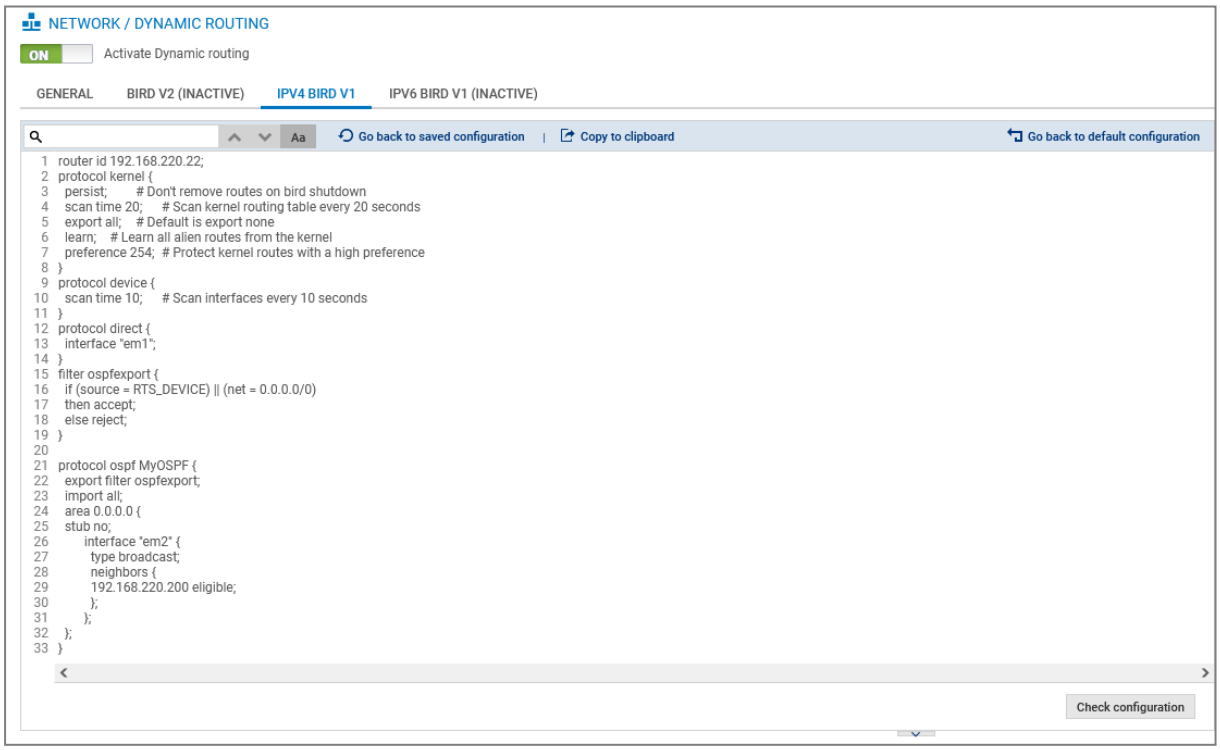
- Dans la partie gauche de l'écran : une trame de configuration BIRD v2 minimaliste comportant les sections de base obligatoires,
- Dans la partie droite de l'écran : la configuration BIRD v1 d'origine du firewall (IPv4 et / ou IPv6).

Il vous permet également de modifier la configuration BIRD v2 du firewall et de la valider.



L'onglet BIRD v1 IPv4

Cet onglet affiche la configuration d'origine du firewall pour le routage dynamique IPv4 géré par BIRD v1. Il vous permet également de la modifier et de la valider.



L'onglet optionnel BIRD v1 IPv6

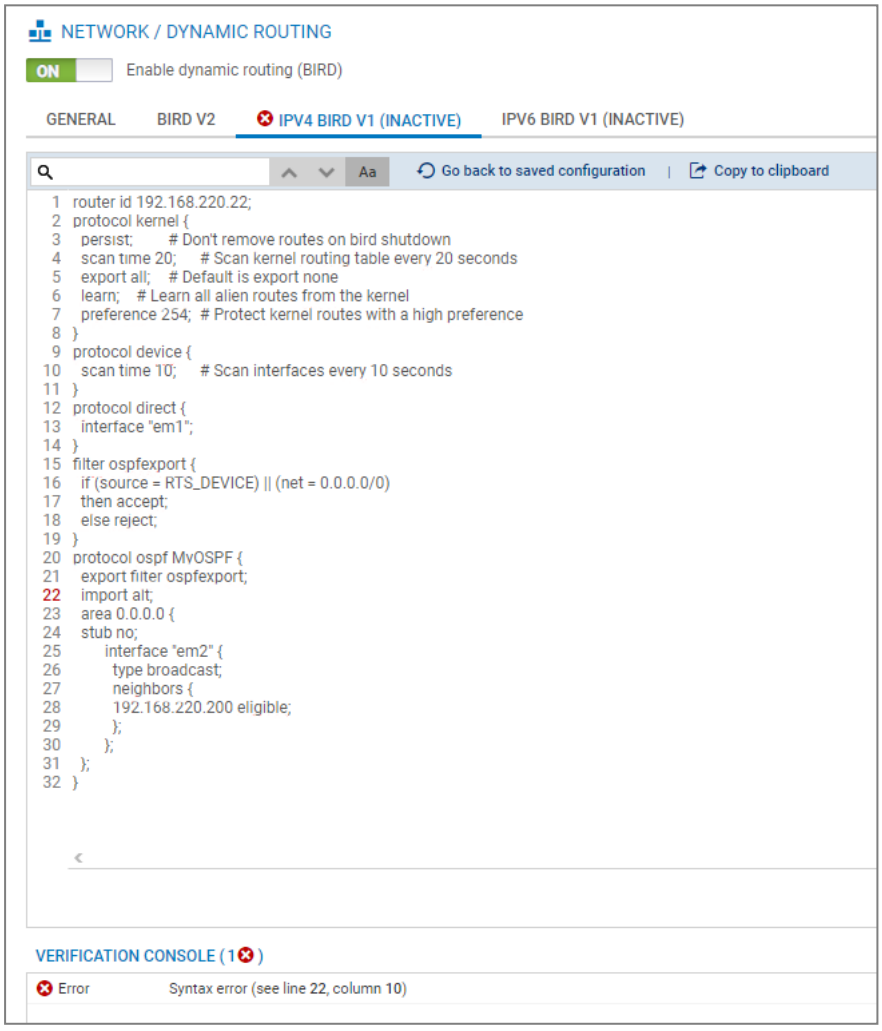
Cet onglet affiche la configuration d'origine du firewall pour le routage dynamique IPv6 géré par BIRD v1. Il vous permet également de la modifier et de la valider.

Sa présentation est identique à celle de l'onglet **BIRD v1 IPv4** / **BIRD v1 IPv4 (INACTIF)**.

La console de contrôle

Lorsque vous cliquez sur le bouton **Vérifier la configuration** depuis l'un des onglets de configuration BIRD présentés ci-dessus, la console de contrôle située en bas de l'écran affiche les éventuelles erreurs de syntaxe rencontrées.

Les erreurs y sont identifiées par leur numéro de ligne et leur numéro de colonne. Les numéros des lignes contenant des erreurs sont également affichés en rouge dans la configuration :



Connaître l'environnement BIRD / Stormshield Network

Dans une configuration sortie d'usine, le module de routage BIRD n'est pas activé.

Il est possible de faire coexister le routage du firewall Stormshield Network et le routage dynamique BIRD. Par exemple, la zone interne peut être gérée avec un protocole de routage dynamique et la zone externe avec les fonctionnalités de routage du firewall (routage statique, passerelles, routage par règles (PBR), objets routeur).

Pour cela, consultez la section Interagir avec le routage Stormshield Network.

Contrairement à la version BIRD v1 qui utilisait deux fichiers de configuration distincts, la configuration du routage dynamique BIRD v2 pour IPv4 et IPv6 s'effectue au sein d'un unique fichier : **/usr/Firewall/ConfigFiles/Bird/bird.conf**.

Démarrer le routage BIRD v2 depuis l'interface Web d'administration

Pour activer et démarrer le routage BIRD v2 :

1. Placez-vous dans le module **Configuration** > **Réseau** > **Routage dynamique** > onglet **Général**,
2. Placez le curseur Activer le routage dynamique sur **ON**,
3. Dans le cadre **Configuration générale** : sélectionnez le bouton radio **BIRD v2.**

Contrôler le routage dynamique BIRD v2 en mode interactif

BIRD dispose d'un mode interactif : ***birdc*** pour BIRD Client.

Ce mode interactif permet de visualiser les états de BIRD, de tester le bon fonctionnement d'une nouvelle configuration en permettant de revenir en arrière, et de créer une configuration temporaire.

En revanche, ce mode interactif ne permet pas de modifier de manière définitive le fichier de configuration de BIRD.

Depuis la console du firewall, lancez ce mode en appelant **birdc** pour contrôler le routage dynamique.

La première information affichée est la version de BIRD :

```
BIRDv2-VMSNSX09I0390A9>birdc
BIRD 2.15.1 ready.
bird>
```

Commandes « Show »

Le caractère "?" vous permet d'afficher la liste des options disponibles :

```
bird> show ?
show status                                 Show router status
show memory                                 Show memory usage
show protocols [<protocol> | "<pattern>"] Show routing protocols
show interfaces                             Show network interfaces
show route ...                              Show routing table
show symbols ...                            Show all known symbolic names
show babel ...                              Show information about Babel
protocol
```

```
show bfd ...                                    Show information about BFD
protocol
show ospf ...                                   Show information about OSPF
protocol
show rip ...                                    Show information about RIP
protocol
show static [<name>]                            Show details of static protocol
```

Exemple :

Afficher toutes les routes.

```
bird> show route
Table master4:
0.0.0.0/0            unicast [kernel1 09:02:35.632] * (254)
        via 172.20.151.254 on em0(out)
172.16.1.125/32      unicast [kernel1 09:02:35.632] * (254)
        dev lo0(loopback)
192.168.220.0/24     unicast [direct1 09:02:35.632] ! (240)
        dev em2(dmz1)
                     unicast [MyOSPF 09:02:35.732] I (150/10)
[192.168.97.219]
        dev em2(dmz1)
172.20.151.3/32      unicast [kernel1 09:02:35.632] * (254)
        dev lo0(loopback)
192.168.220.21/32    unicast [kernel1 09:02:35.632] * (254)
        dev lo0(loopback)
```

Exemple :

Afficher les routes par instance de protocole. Dans ce cas, l'instance est **MyOSPF**.

```
bird> show route protocol MyOSPF
Table master4:
192.168.220.0/24    unicast [MyOSPF 09:02:35.732] I (150/10)
[192.168.97.219]
       dev em2(dmz1)
bird>
```

Dans ***birdc***, la plupart des commandes sont communes à l'ensemble des protocoles. Ainsi par exemple, les routes annoncées à un voisin BGP sont visualisées par une commande qui fait appel au filtre d'export (filtre nommé *ospfexport* dans cet exemple) :

```
bird> show route filter ospfexport
Table master4:
0.0.0.0/0            unicast [kernel1 09:02:35.632] * (254)
       via 172.20.151.254 on em0(out)
192.168.220.0/24     unicast [direct1 09:02:35.632] ! (240)
       dev em2(dmz1)
```

Debug

Les commandes *Show* donnent de nombreux renseignements sur les instances. Elles permettent de diagnostiquer les problèmes, qu'ils soient dus à une mauvaise configuration, un problème de réseau, ou autre.

```
bird> show protocol all router1
Name       Proto      Net Type   Table      State  Since         Info
router1    BGP        Undefined  ---        start  14:11:41.925  Active
    Socket: Connection closed
       Description:    My 1st BGP uplink
       BGP state:          Active
```

```
                Neighbor address: 100.100.100.100
                Neighbor AS:      65001
                Local AS:         65065
                Connect delay:    1.041/5
                Last error:       Socket: Connection closed
        Channel ipv4
                State:          DOWN
                Table:          master4
                Preference:     100
                Input filter:   ACCEPT
                Output filter:  (unnamed)
                IGP IPv4 table: master4
```

Pour activer la réception des messages systèmes sur la console, entrez la commande echo all puis echo off pour stopper ces logs.

```
bird> echo all
bird> >>> router1: Connecting to 100.100.100.100 from local address
200.200.200.200
>>> router1: Socket error: bind: Can't assign requested address
>>> router1: Connection closed
>>> router1: Connect delayed by 5 seconds
```

Les événements de *debug* sont visualisés globalement ou par exemple par instance de protocole. L'exploitation des commandes de *debug* est un outil intéressant qui complète efficacement les commandes de visualisation d'états.

```
bird> debug ospf_router2_v4 all
bird> echo all
>>> ospf_router2_v4 < added 0.0.0.0/0 via 192.168.97.1 on em0
>>> ospf_router2_v4 < replaced 100.100.100.100/32 via 192.168.97.101 on
em0
>>> ospf_router2_v4 > updated 1.1.1.0/24 via 192.168.97.1 on em0
>>> ospf_router2_v4 < rejected by protocol 1.1.1.0/24 via 192.168.97.1 on
em0
>>> ospf_router2_v4 > updated [best] 1.1.1.0/24 via 192.168.97.1 on em0
>>> ospf_router2_v4 < replaced 2.2.2.0/24 via 192.168.97.101 on em0
>>> ospf_router2_v4 < replaced 2.2.4.0/24 via 192.168.97.101 on em0
```

Test temporaire d'une nouvelle configuration

On souhaite tester une nouvelle configuration **bird_conf_to_test.conf**. Pour cela, activez BIRD en utilisant une configuration **bird.conf** dont le fonctionnement est validé, puis lancez le mode interactif **birdc** depuis la console du firewall..

Pour vérifier la syntaxe du fichier sans l'appliquer :

```
bird> configure check "bird_conf_to_test.conf"
```

Appliquez ensuite temporairement cette configuration pendant 60 secondes par la commande :

```
bird> configure "bird_conf_to_test.conf" timeout 60
```

La nouvelle configuration s'applique. Si le firewall n'est plus joignable ou sans confirmation de la part de l'administrateur, la configuration précédente sera ré-appliquée automatiquement au bout de 60 secondes.

Si la nouvelle configuration est considérée comme valide, on peut la confirmer grâce à :

```
bird> configure confirm
```

Si la nouvelle configuration n'est pas validée et que le firewall est encore joignable, on peut revenir en arrière immédiatement grâce à :

```
bird> configure undo
```

Configurer le routage dynamique

La configuration de BIRD v2 est réalisée dans le module **Configuration** > **Réseau** > **Routage dynamique** > onglet **BIRD v2 (INACTIF)**.

Toute mise en œuvre demande à minima à ce que les lignes suivantes soient configurées afin de définir un environnement basique de coopération avec le système.

```
# WARNING : There is no implicit filtering rules implemented in SNS which
allow to use various BIRD protocols
#           There is more information in Technical Note > Bird Dynamic
Routing
# Enable extra logs
#     Possible values are :
#         off No extra log
#         all All extra log
#         adj Log neighbors state changes
#         route Log route addition/deletion
#         or a combination of adj and route separated by '|'
#         IE. sns_log adj|route;
sns_log off;    # default is "no extra log"
# This pseudo-protocol watches all interface up/down events.
protocol device {
       scan time 10;        # Scan interfaces every 10 seconds
}
# The direct protocol automatically generates device routes to
# all network interfaces.
protocol direct {
       ipv4;                # Minimal IPv4 default channel config
}
# This pseudo-protocol performs synchronization between BIRD's routing
# tables and the kernel.
protocol kernel {
       learn;               # Learn all alien routes from the kernel
       persist;             # Don't remove routes on bird shutdown
       scan time 20;        # Scan kernel routing table every 20 seconds
       ipv4 {
               import all;      # Default is import all
               export none;     # THIS CONFIGURATION MUST BE ADJUSTED
               preference 254; # Protect existing routes
       };
}
# This pseudo-protocol is used to configure static routes.
protocol static MyStaticRoutes {
       ipv4;
#    route 0.0.0.0/0 via 173.1.1.1;            # Default route
#    route 192.168.250.0/24 via "out";         # Declare network via
username interface
#    route 192.168.251.0/24 via 173.1.1.2;     # Declare network via
specific gateway
#    route 192.168.252.0/24 via 173.1.1.3 bfd; # Declare network via
specific gateway with BFD enabled
}
```

Nous ne nous attardons pas ici sur le détail de chaque ligne de configuration. Si vous désirez en obtenir des explications exhaustives, consultez la documentation en ligne de BIRD à l'adresse :

http://bird.network.cz/?get_doc&f=bird.html (anglais uniquement).

Les notions les plus importantes sont celle d'instance de protocole et celle de filtre.

Une instance de protocole peut être soit BGP soit RIP soit OSPF et définit une configuration appropriée. Vous pouvez éventuellement définir plusieurs instances pour un même protocole.

Chaque instance de protocole est connectée à une table de routage interne à BIRD. Cette connexion est contrôlée par deux filtres qui peuvent accepter, refuser ou modifier les routes.

Le filtre d'exportation contrôle les routes transmises de la table de routage interne à BIRD vers le protocole. Le filtre d'importation fait de même dans l'autre sens.

IMPORTANT
Il est nécessaire d'être précis lors de la mise en œuvre d'un filtrage de routes.
L'utilisation d'export ou d'import complets de routes (par exemple, import all;) entre instances de protocole peut avoir des effets destructeurs.

Connaître les règles de syntaxe

- Le texte sur la ligne placé après # est un commentaire,
- Le texte entouré de /* et */ est un commentaire,
- Les blocs de plusieurs options sont placés entre accolades {},
- Chaque option se termine par un point-virgule ;,
- La configuration est sensible à la casse.

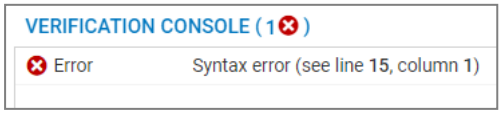
Vérifier une configuration

L'exemple décrit ci-dessous indique comment vérifier une configuration présentant deux erreurs de syntaxe.

La configuration suivante est saisie dans la fenêtre de configuration :

```
# WARNING : There is no implicit filtering rules implemented in SNS which
allow to use various BIRD protocols
#           There is more information in Technical Note > Bird Dynamic
Routing
sns_log off;    # default is "no extra log"

router id 192.168.97.219;
protocol kernel {
persist;              # Don't remove routes on bird shutdown
scan time 20;         # Scan kernel routing table every 20 seconds
ipv4 {
export all;     # THIS CONFIGURATION MUST BE ADJUSTED
preference 254; # Protect existing routes
};
learn;                # Learn all alien routes from the kernel

protocol device {
scan time 10        # Scan interfaces every 10 seconds
}
```

Cliquez sur le bouton **Vérifier la configuration**. La console de vérification située au bas de l'écran indique la première erreur rencontrée.

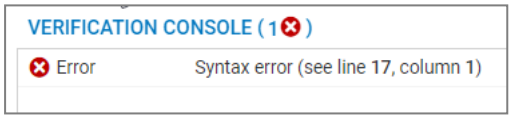


Le numéro de ligne 15 s'affiche également en rouge dans la configuration.

```
1  sns_log off;   # default is "no extra log"
2
3  router id 192.168.97.219;
4
5  protocol kernel {
6    persist;       # Don't remove routes on bird shutdown
7    scan time 20;     # Scan kernel routing table every 20 seconds
8    ipv4 {
9    export all;    # Default is export none
10   preference 254;  # Protect kernel routes with a high preference
11   };
12   learn;    # Learn all alien routes from the kernel
13
14
15 protocol device {
16   scan time 10      # Scan interfaces every 10 seconds
17 }
18
```

Explication : si une accolade de fermeture de bloc est oubliée, l'erreur mentionne **la première ligne du bloc suivant**, ligne ne correspondant pas à une commande autorisée du bloc non fermé. Il faut donc insérer le caractère « **}** » en fin de ligne précédente.

Une fois cette première erreur corrigée, cliquez de nouveau sur le bouton **Vérifier la configuration** pour afficher l'erreur de syntaxe suivante dans la console de vérification (avec le numéro de ligne correspondant à l'erreur surligné en rouge dans la configuration) :




```
1  sns_log off;   # default is "no extra log"
2
3  router id 192.168.97.219;
4
5  protocol kernel {
6    persist;       # Don't remove routes on bird shutdown
7    scan time 20;     # Scan kernel routing table every 20 seconds
8    ipv4 {
9    export all;    # Default is export none
10   preference 254;  # Protect kernel routes with a high preference
11   };
12   learn;    # Learn all alien routes from the kernel
13   }
14
15 protocol device {
16   scan time 10      # Scan interfaces every 10 seconds
17 }
18
```

Explication : il faut insérer le caractère « **;** » à la fin de la ligne précédente (ligne 16 sur cet exemple).

Interagir avec le routage Stormshield Network

Grâce à la configuration fournie par défaut sur les firewalls Stormshield Network, le routage du firewall est prioritaire sur le routage dynamique (préférence maximale de 254).

ATTENTION
Pendant la reconfiguration des routes du firewall, celles-ci sont temporairement effacées et BIRD peut alors configurer ses propres routes. Vous devez protéger le routage du firewall grâce à un filtre d'export sur le pseudo-protocole *kernel*.

Voici un exemple de filtre qui protège la route par défaut et la route statique 1.2.3.0/24 :

```
filter protect_Stormshield_routes{
        if (net = 0.0.0.0/0) || (net = 1.2.3.0/24) the reject;
        else accept;
}
protocol kernel {
        learn;              # Learn all alien routes from the kernel
        persist;            # Don't remove routes on bird shutdown
        scan time 20;       # Scan kernel routing table every 20 seconds
        ipv4 {
                import all;     # Default is import all
                export filter protect_Stormshield_routes;
                preference 254; # Protect existing routes
        };
}
```

Routage dynamique prioritaire sur le routage Stormshield Network

Dans la table du routage dynamique BIRD, si vous voulez que le routage dynamique soit prioritaire sur le routage Stormshield Network, il faut que les routes obtenues par routage dynamique (protocole BGP, OSPF ou RIP) aient une valeur de préférence plus élevée que les routes obtenues par le système (pseudo-protocole *kernel*). En revanche, cela n'impacte pas la table de routage du firewall lui-même (affichable par la commande `netstat -r`).

Vous devez donc diminuer la valeur de préférence de *kernel*, par exemple à 1 :

```
protocol kernel {
        (...)
        ipv4 {
        (...)
                preference 1; # Protect existing routes
        };
}
```

Routage des interfaces du firewall

Si les interfaces du firewall sont configurées avec des sous-réseaux différents, et que vous souhaitez transmettre les sous-réseaux des interfaces via BIRD, utilisez le pseudo-protocole *direct*.

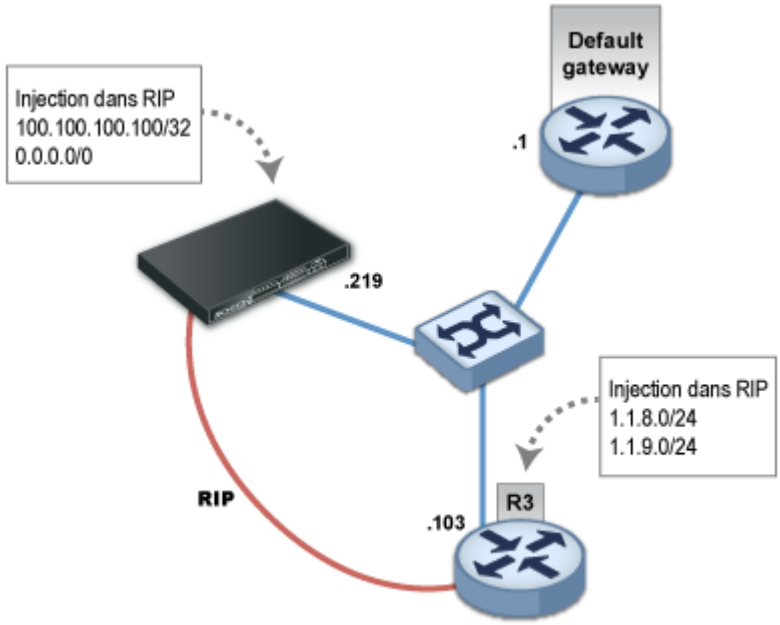
Par défaut, toutes les interfaces sont prises en compte. Vous pouvez restreindre l'ensemble des interfaces prises en compte grâce à l'attribut *interface*.

```
protocol direct {
        interface "-vlan*","*";
}
```

Comprendre des configurations simples

RIP

La version supportée est RIP v2.
Voici ci-dessous la configuration « RIP simple ».



On configure une route par défaut et une route statique vers 100.100.100.100/32 :

STATIC ROUTES

Searching... + Add × Delete

Status	Destination network (host, network or group ...	Interface	Address range	Protected	Gateway
on	u500s_ebgp	dmz4	100.100.100.100		u500s_priv

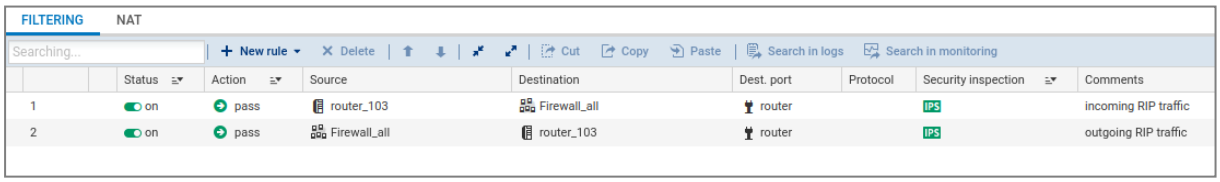
On configure RIP v2 en spécifiant « multicast » comme mode RIP associé à l'interface « em0 ».

```
sns_log off;   # default is "no extra log"
router id  192.168.97.219;

# This pseudo-protocol performs synchronization
# between BIRD's routing tables and the kernel.
protocol kernel {
       persist;                # Don't remove routes on bird shutdown
       scan time 20;           # Scan kernel routing table every 20 seconds
       learn;                  # Learn all alien routes from the kernel
       ipv4 {
               export all;     # Default is export none
               preference 254; # Protect kernel routes with high preference
       };
}

# This pseudo-protocol watches all interface up/down events.
protocol device {
       scan time 10;       # Scan interfaces every 10 seconds
```

```
}

# The direct protocol automatically
# generates device routes to all network interfaces.
protocol direct {
       ipv4;               # Minimal IPv4 default channel config
}

filter  ripexport {
       if (net = 0.0.0.0/0) || (net = 100.100.100.100/32)
       then accept;
       else reject;
}

protocol rip MyRIP {
       debug all;
       interface "em0" {
       mode multicast;
       authentication none;
       };
       ipv4 {
               import all;
               export filter ripexport;
       };
}

# This pseudo-protocol is used to configure static routes.
protocol static MyStaticRoutes {
       ipv4;
}
```

Autoriser le protocole RIP dans la politique de filtrage

Des règles de filtrage sont nécessaires pour autoriser les flux de routage RIP vers et depuis le firewall :



Vérifier le bon fonctionnement du routage dynamique RIP

Afficher l'état de l'instance de protocole :

```
bird> show protocols all MyRIP
Name    Proto    Net Type         Table    State    Since           Info
MyRIP   RIP      ipv4             master4  up       10:08:56
  Channel ipv4
    State:             UP
    Table:             master4
    Preference:        120
    Input filter:      ACCEPT
    Output filter:     ripexport
    Routes:    0 imported, 1 exported, 0 preferred
    Route change stats:  received rejected filtered ignored accepted
      Import updates:    0        0        0        0       0
      Import withdraws:  7        0        ---      7       0
      Export updates:    15       0        13       ---     2
```

```
      Export withdraws:  3        ---      ---      ---     1
```

Afficher les routes apprises :

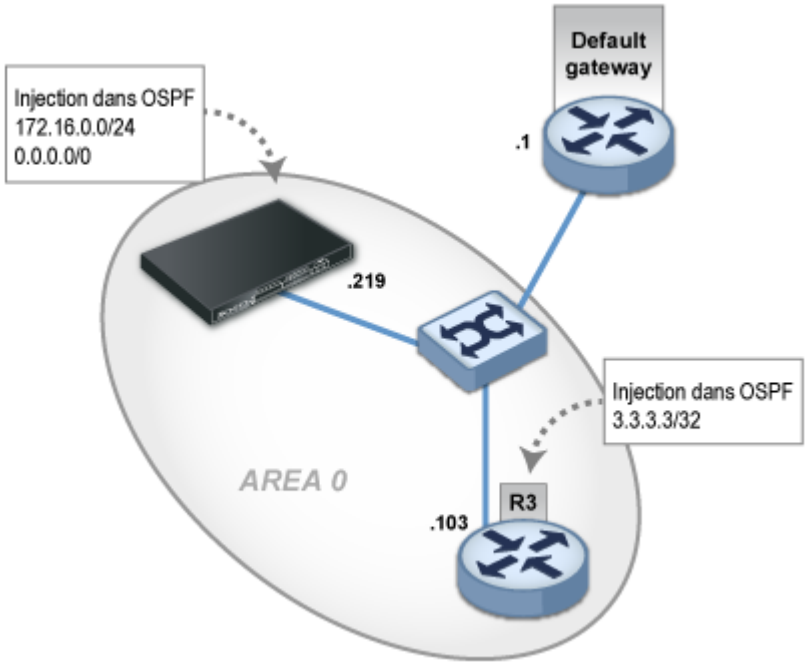
```
bird> show route primary protocol MyRIP
192.168.97.0/24 via 10.200.45.250 on eth0 [MyRIP 10:29:19] ! (120/2)
1.1.9.0/24 via 10.200.45.250 on eth0 [MyRIP 10:29:19] * (120/2)
1.1.8.0/24 via 10.200.45.250 on eth0 [MyRIP 10:29:19] * (120/2)
```

Voici ci-dessous les routes reçues par le voisin. Notez que la route par défaut est reçue. L'export de cette route est en effet normalement rejeté par les routeurs du marché. Ici, il est nécessaire de le filtrer explicitement.

```
bird> show route primary protocol MyRIP
0.0.0.0/0 via 192.168.97.219 on eth0 [MyRIP 10:36] * (120/2)
100.100.100.100/32 via 192.168.97.101 on eth0 [MyRIP 10:36 from
192.168.97.219] * (120/2)
```

OSPF

Les versions supportées sont OSPF v2 pour IPv4 et OSPF v3 pour IPv6.
Voici ci-dessous la configuration « OSPF_simple » :



Elle consiste à déployer une aire 0 sur un LAN où l'on désigne explicitement un voisin éventuel. Toutes les routes sont importées d'OSPF. On redistribue dans OSPF la route du sous-réseau directement relié à l'interface em3 (172.16.0.0/24), ainsi que la route par défaut.

```
sns_log off;   # default is "no extra log"

router id 192.168.97.219;

# This pseudo-protocol performs synchronization
# between BIRD's routing tables and the kernel.
protocol kernel {
       learn;              # Learn all alien routes from the kernel
```

```
        persist;            # Don't remove routes on bird shutdown
        scan time 20;       # Scan kernel routing table every 20 seconds
        ipv4 {
                export all;     # THIS CONFIGURATION MUST BE ADJUSTED
                preference 254; # Protect existing routes
        };
}

# This pseudo-protocol watches all interface up/down events.
protocol device {
        scan time 10;       # Scan interfaces every 10 seconds
}

# The direct protocol automatically generates
# device routes to all network interfaces.
protocol direct {
        ipv4;               # Minimal IPv4 default channel config
        interface "em2";
}

filter ospfexport {
        if (source = RTS_DEVICE) || (net = 0.0.0.0/0)
        then accept;
        else reject;
}

protocol ospf MyOSPF {
area 0.0.0.0 {
        stub no;
        interface "em2" {
        type broadcast;
        neighbors {
                192.168.97.103 eligible;
        };
        };
        };
        ipv4 {
                export filter ospfexport;
                import all;
        };
}
```

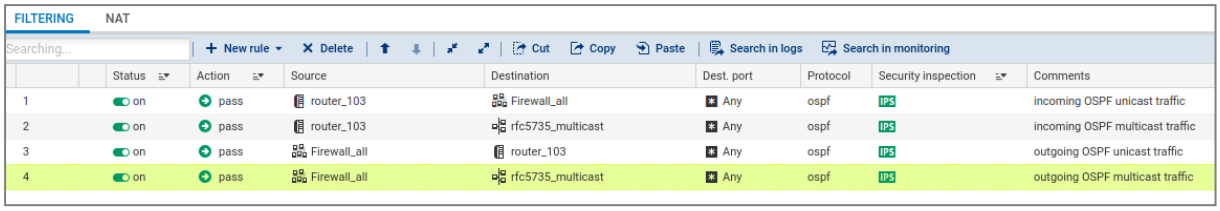
NOTE
Il est conseillé de positionner le paramètre "`priority 0`" dans la section *interface* de la configuration du noeud OSPF afin de désactiver la participation du firewall aux élections pour les rôles de Designated Router / Backup Designated Router.

Autoriser le protocole OSPF dans la politique de filtrage

Des règles de filtrage sont nécessaires pour autoriser les flux de routage OSPF vers et depuis le firewall.

Dans l'exemple de politique de filtrage suivant, l'objet *router_103* représente l'adresse IP (192.168.97.103) du voisin OSPF déclaré explicitement dans la configuration du firewall.

IMPORTANT
Pour le bon fonctionnement d'OSPF, vous devez autoriser le trafic unicast d'OSPF en plus du trafic multicast comme illustré dans cet exemple de politique de filtrage.



Vérifier le bon fonctionnement du routage dynamique OSPF

La commande suivante indique que le voisinage est bien établi (indiqué par l'état « full »). Le voisin est déclaré comme « Designated Router » (indiqué par l'état « dr ») :

```
bird> show ospf neighbors
MyOSPF:
Router ID               Pri      State    DTime    Interface    Router IP
192.168.97.103          1        full/dr  00:34    em4          192.168.97.103
```

Routes reçues:

```
bird> show route protocol MyOSPF
3.3.3.3/32 via 192.168.97.103 on em4 [MyOSPF 16:17:38] * E2 (150/10/10000)
[192.168.97.103]
192.168.97.0/24 dev em4 [MyOSPF 16:15:43] * I (150/10) [192.168.97.219]
```

On peut afficher la topologie OSPF :

```
bird> show ospf topology
area 0.0.0.0
       router 192.168.97.103
              distance 10
              network 192.168.97.0/24 metric 10
       router 192.168.97.219
              distance 0
              network  192.168.97.0/24 metric 10
       network 192.168.97.0/24
              dr 192.168.97.103
              distance 10
              router 192.168.97.103
              router 192.168.97.219
```

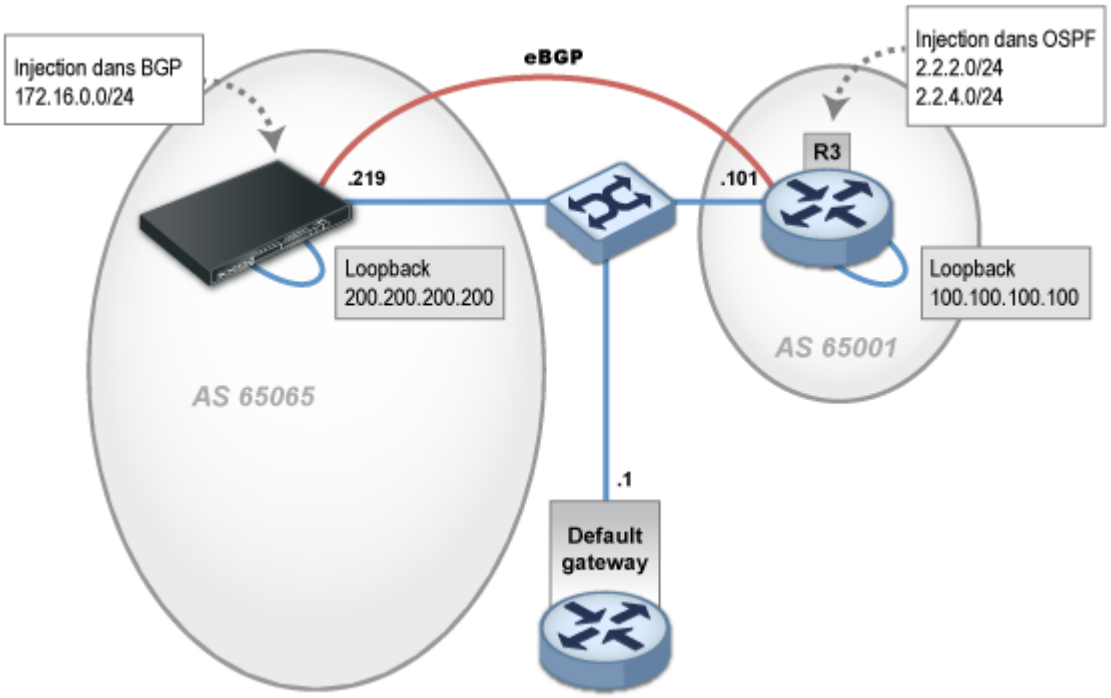
Ainsi que la base de données LSA :

```
bird> show ospf lsadb
Global
Type   LS ID           Router          Age     Sequence        Checksum
0005   3.3.3.3         192.168.97.103  501     8000000a        ec8a
0005   172.16.0.255    192.168.97.219  1150    80000001        81b6
0005   0.0.0.0         192.168.97.219  1150    80000001        37f1
Area   0.0.0.0
Type   LS ID           Router          Age     Sequence        Checksum
0001   192.168.97.103  192.168.97.103  455     8000000a        2254
0002   192.168.97.103  192.168.97.103  456     80000006        9384
0001   192.168.97.219  192.168.97.219  1144    8000041b        0bf8
```

NOTE
Notez que le type de LSA est présenté à gauche alors qu'il sert généralement de délimiteur horizontal dans les conventions d'affichage traditionnelles.

BGP

La version supportée est BGP v4 pour IPv4 et IPv6.
Voici ci-dessous la configuration « BGP simple » :



La configuration « BGP_simple » est implémentée de la façon suivante :

```
sns_log off;   # default is "no extra log"

router id 192.168.97.219;
protocol kernel {
       persist;          # Don't remove routes on bird shutdown
       scan time 20;     # Scan kernel routing table every 20 seconds
       ipv4 {
               export all;      # Default is export none
               preference 254;  # Protect kernel routes with high preference
       };
       learn;            # Learn all alien routes from the kernel
}

protocol  device {
       scan time 10;     # Scan interfaces every 10 seconds
}

protocol  direct {
       interface "em3";
}

protocol bgp MyBGP {
       description  "My 1st BGP uplink";
       local as 65065;
       neighbor 100.100.100.100 as 65001;
       multihop 5;
       hold time 180;
       keepalive time 60;
       ipv4 {
```

```
                import all;
                export where source = RTS_DEVICE;
        };
        default bgp_local_pref 100;
        source address  200.200.200.200;
}

# This pseudo-protocol is used to configure static routes.
protocol static MyStaticRoutes {
        ipv4;
}
```

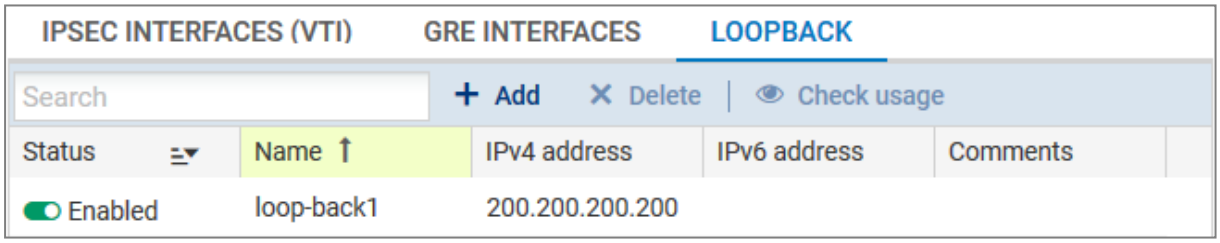
Explications

Contrairement à la majorité des routeurs du marché, il est nécessaire de spécifier l'AS local pour chaque instance BGP.

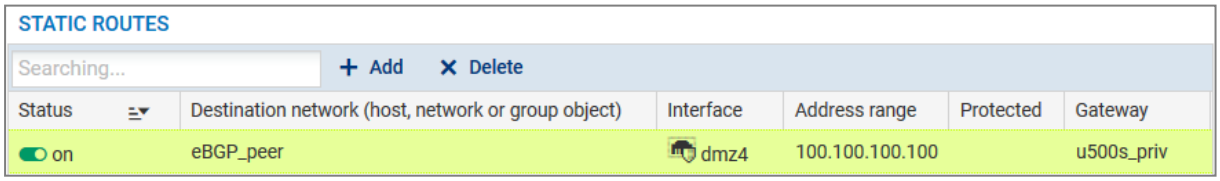
Selon les bonnes pratiques, on monte cette session eBGP entre des interfaces *loopbacks* et non pas les interfaces physiques. Il est donc nécessaire de configurer l'IP de la *loopback* locale en question (200.200.200.200/32), de spécifier cette adresse comme source et une route statique vers la *loopback* du voisin.

Interfaces virtuelles Loopback

L'interface d'administration web permet de configurer les interfaces de type loopback via le module **Configuration** > **Réseau** > **Interfaces virtuelles**, onglet *Loopback :*



Il est conseillé de déclarer la route statique vers la loopback distante sur le firewall en dehors de la configuration BIRD, via le module **Configuration** > **Réseau** > **Routage**, onglets *Routes statiques*, afin d'éviter que le trafic BGP soit bloqué par des alarmes "Usurpation d'adresse IP" :



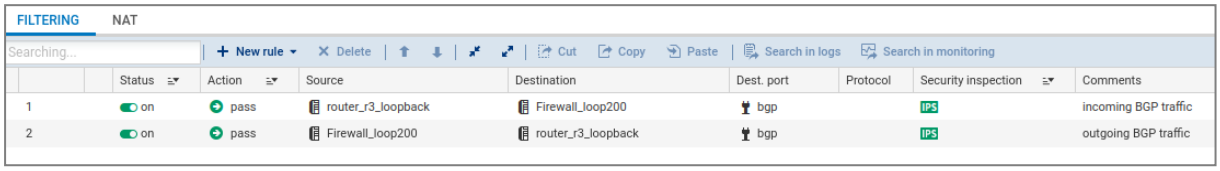
A nouveau, on sélectionne seulement le sous-réseau 172.16.0.0/24 relié directement à l'interface *em3* comme route à annoncer à nos voisins.

Ici on a défini un filtre d'export anonyme, directement dans l'instruction "export", grâce au mot-clé "where". Ce filtre d'export sélectionne les routes dont la source est RTS_DEVICE, c'est-à-dire les routes obtenues par le pseudo-protocole direct.

La valeur du *hold-time* est spécifiée à 180s, valeur habituelle du marché. BIRD implémente par défaut 240s. Il n'est pas nécessaire de spécifier la valeur du délai de *keepalive* (calculé à 1/3 du *hold-time*) mais nous le mentionnons explicitement pour plus de lisibilité. De même pour la *local-preference* par défaut.

Autoriser le protocole BGP dans la politique de filtrage

Des règles de filtrage sont nécessaires pour autoriser les flux de routage BGP vers et depuis le firewall :



Vérifier le bon fonctionnement du routage dynamique BGP

La commande « show protocols » ci-dessous confirme que la session est bien fonctionnelle.

```
bird> show protocols router1
name            proto    table    state    since    info
router1         BGP      master   up       12:47    Established
```

Les routes sont bien reçues du voisin :

```
bird> show route protocol router1
100.100.100.100/32 via 192.168.97.101 on em0 [router1 13:09 from
100.100.100.100]
 (100/?) [AS65001?]
2.2.2.0/24         via 192.168.97.101 on em0 [router1 13:09 from
100.100.100.100]
 *(100/?)[AS65001?]
2.2.4.0/24         via 192.168.97.101 on em0[router1 13:09 from
100.100.100.100]
 *(100/?)[AS65001?]
```

Sur le voisin BGP on reçoit bien la route annoncée et libérée par le filtre. La route 1.1.1.1/32 est pour sa part, effectivement bloquée.

Authentification

Il est possible de mettre en œuvre une authentification TCP-MD5 entre routeurs BGP au sein de BIRD.

Cette méthode permet ainsi la protection des sessions BGP par authentification des trames dans l'entête TCP conformément à la RFC2385.

Cela se traduit par l'ajout de la directive *"password"* dans la configuration du routeur BGP au sein des fichiers **/usr/Firewall/ConfigFiles/Bird/bird.conf** (routage dynamique des paquets IPv4 et IPv6). Vous devez également impérativement ajouter la directive "source address" et préciser l'adresse IP de l'interface utilisée pour réaliser cette authentification.

Par exemple :

```
protocol bgp MyBGP {
       description  "My 1st BGP uplink";
       local as 65065;
       neighbor 100.100.100.100 as 65001;
       password "very_secret";
       multihop 5;
       hold time 180;
       keepalive time 60;
       ipv4 {
```

```
                import all;
                export where source = RTS_DEVICE;
        };
        default bgp_local_pref 100;
        source address  200.200.200.200;
}
```

NOTE
Les mots de passe ne doivent pas contenir d'espace ni de signe égal ('=').

Cas d'une configuration en haute disponibilité (cluster)

En cas d'utilisation du protocole de routage dynamique BGP sur un cluster de firewalls SNS, et pour permettre au voisin BGP de fermer proprement une session BGP en cas de bascule au sein du cluster, il est utile de définir une instance BFD dans la configuration BIRD :

```
protocol bfd mybfdsession {
        neighbor myneighborip;
}
```

Dans cet exemple :

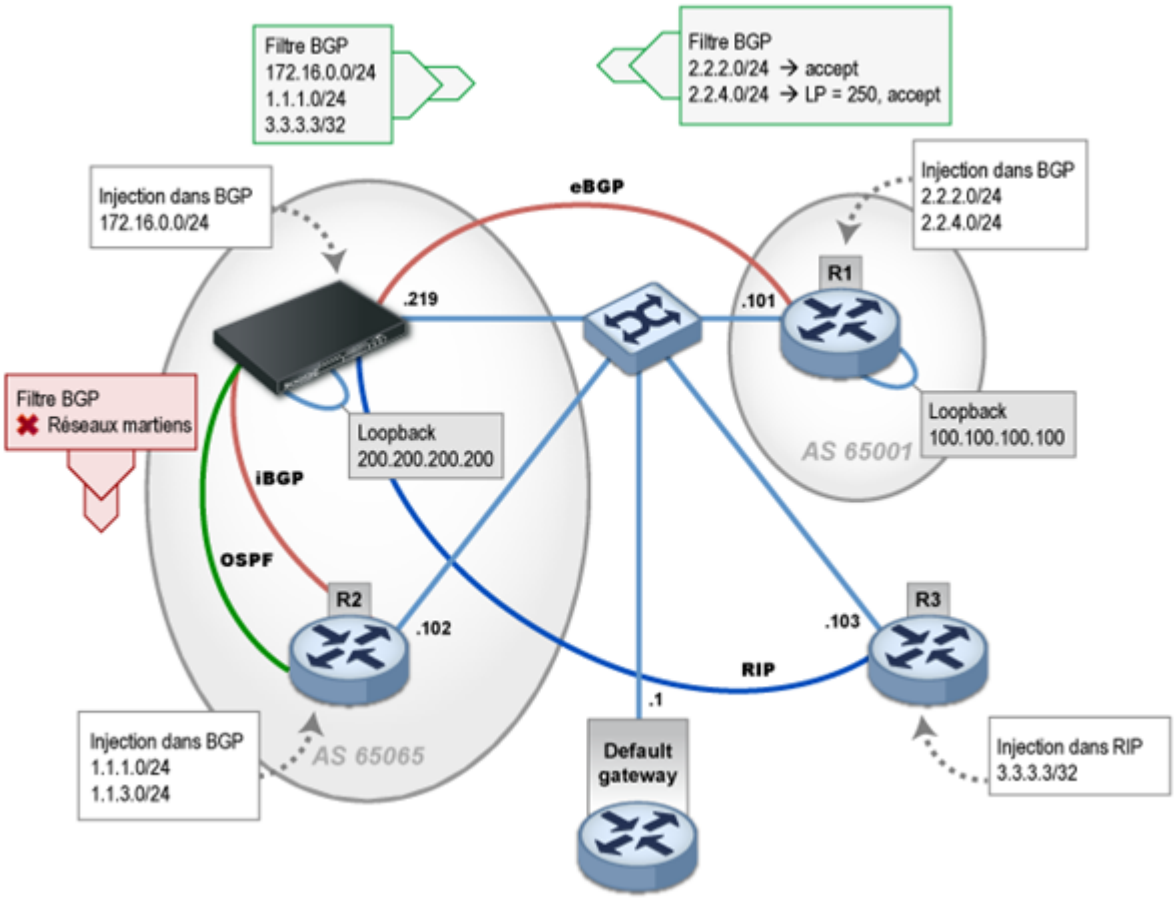
```
protocol bfd mybfdsession {
        neighbor 100.100.100.100;
}
```

Et d'appeler cette instance dans la configuration BGP de la manière suivante :

```
protocol bgp MyBGP {
…
        bfd graceful;
        connect retry time 5;
…
}
```

Configuration avancée

On met ici en œuvre la configuration avancée. Cette configuration réunit les trois configurations simples et comporte en plus une liaison iBGP établie en parallèle de la liaison OSPF.



Le réseau du client comprend le routeur R2, R3 et le firewall Stormshield Network. Le routeur R1 est un voisin BGP externe. Ce réseau représente un cas réaliste d'architecture, à l'exception du fait que tous les routeurs sont connectés physiquement par le biais d'un LAN unique.

On met en œuvre une politique standard de filtrage pour :

- N'annoncer que les réseaux publics en BGP vers l'extérieur,
- Ne pas propager de réseaux internes ou martians dans BGP interne,
- Tagguer une des routes apprises en eBGP avec une local-preference de 250 ; cette mesure est généralement mise en œuvre pour contrôler le partage de charge entre plusieurs voisins eBGP,
- N'annoncer dans OSPF qu'une route par défaut,
- N'annoncer dans RIP qu'une route par défaut.

Les réseaux annoncés par R2 et R3 le sont respectivement via BGP et RIP. L'utilisation d'OSPF pour annoncer la route par défaut n'a qu'une utilité pédagogique.

Configuration BIRD

Ci-dessous le fichier de configuration équivalent en BIRD :

```
router id 192.168.97.219;

function is_locormartians()
       prefix set martians;
       {
       martians = [ 169.254.0.0/16+, 172.16.0.0/12+,
192.168.0.0/16+,10.0.0.0/8+, 224.0.0.0/4+, 240.0.0.0/4+ ];
       # default
       if net.ip = 0.0.0.0 then return true;
       # LIR not authorized
       if (net.len < 8) || (net.len > 24) then return true;
       # martians
       if net ~ martians then return true;
       # local
       if net = 100.100.100.100/32 then return true;
       return false;
}

filter out_eBGP {
       if net ~ [ 172.16.0.0/24, 3.3.3.3/32, 1.1.1.0/24 ]
       then accept;
       else reject;
}

filter out_iBGP {
       if ( is_locormartians() )
       then reject;
       else accept;
}

filter lp_tag_in {
       if net = 2.2.4.0/24 then {
       bgp_local_pref = 250;
       accept;
       } else accept;
}

filter default_ok {
       if net = 0.0.0.0/0 then {
       accept;
       } else reject;
}

sns_log all;   # default is "no extra log"

# This pseudo-protocol watches all interface up/down events.
protocol device {
       scan time 10;       # Scan interfaces every 10 seconds
}


# The direct protocol automatically generates device routes to
# all network interfaces.
protocol direct {
       interface "em3";
       ipv4;
}

# This pseudo-protocol performs synchronization between BIRD's routing
# tables and the kernel.
```

```
protocol kernel {
        learn;              # Learn all alien routes from the kernel
        persist;            # Don't remove routes on bird shutdown
        scan time 20;       # Scan kernel routing table every 20 seconds
        ipv4 {
                export all;
                preference 254; # Protect existing routes
        };
}

protocol rip MyRIP {
# You can also use an explicit name
        debug all;
        interface "em4" {
                mode multicast;
                authentication none;
        };
        ipv4 {
                import all;
                export filter default_ok;
        };
}

protocol ospf MyOSPF {
        area 0.0.0.0 {
        stub no;
        interface "em4" {
                type broadcast;
        };
        };
        ipv4 {
                export filter default_ok;
                import all;
        };
}

protocol bgp router1 {
        debug all;
        description "My 1st BGP uplink";
        local as 65065;
        neighbor 100.100.100.100 as 65001;
        source address 200.200.200.200;
        multihop 5;
        hold time 180;
        keepalive time 60;
        ipv4{
                export filter out_eBGP;
                import filter lp_tag_in;
        };
}

protocol bgp router2 {
        description "My local BGP neighbor";
        local as 65065;
        neighbor 192.168.97.102 as 65065;
        keepalive time 60;
        ipv4 {
                next hop self;
                export filter out_iBGP;
                import all;
        };
}
```

NOTE
Il est conseillé de positionner le paramètre "`priority 0`" dans la section *interface* de la configuration du noeud OSPF afin de désactiver la participation du firewall aux élections pour les rôles de Designated Router / Backup Designated Router.

Autoriser les protocoles RIP, BGP et OSPF dans la politique de filtrage

En vous référant aux exemples de politique de filtrage présentés dans les configuration simples RIP, BGP et OSPF, ajoutez des règles de filtrage pour autoriser les flux de routage RIP, BGP et OSPF vers et depuis le firewall.

Vérifier le bon fonctionnement du routage dynamique

Table de routage du firewall Stormshield Network

```
bird> show route
0.0.0.0/0           via 192.168.97.1 on em4 [kernel1 14:37:15] * (254)
100.100.100.100/32 via 192.168.97.101 on em4 [kernel1 14:37:15] * (254)
3.3.3.3/32          via 192.168.97.103 on em4  [MyRIP 14:37:06] * (120/2)
192.168.97.0/24     dev em4 [MyOSPF 14:01:33] * I (150/10) [192.168.97.102]
                    via 192.168.97.102 on em4 [router2 14:01:17] (100/10) [i]
1.1.1.0/24          via 192.168.97.102 on em4 [MyOSPF 14:01:36] * E2
(150/10/10000) [192.168.97.102]
                    via 192.168.97.102 on em4 [router2 14:01:17] (100/10) [i]
1.1.3.0/24          via 192.168.97.102 on em4 [MyOSPF 14:01:36] * E2
(150/10/10000) [192.168.97.102]
                    via 192.168.97.102 on em4 [router2 14:01:17] (100/10) [i]
2.2.2.0/24          via 192.168.97.101 on em4 [router1 13:54:12 from
100.100.100.100] * (100/?) [AS65001i]
2.2.4.0/24          via 192.168.97.101 on em4 [router1 14:01:17 from
100.100.100.100] * (100/?) [AS65001i]
172.16.0.254/32     dev lo0 [kernel1 14:37:15] * (254)
192.168.97.219/32   dev lo0 [kernel1 14:37:15] * (254)
172.16.0.0/24       dev em3 [direct1 13:54:11] * (240)
10.200.45.254/32    dev lo0 [kernel1 14:37:15] * (254)
```

Afin de pouvoir vérifier la local-preference sur la route 2.2.4.0/24 on affiche le détail des routes de l'instance du protocole router1 :

```
bird> show route protocol router1 all
2.2.2.0/24 via 192.168.97.101 on em4 [router1 13:54:12 from
100.100.100.100] * (100/?) [AS65001i]
 Type: BGP unicast univ
 BGP.origin: IGP
 BGP.as_path: 65001
 BGP.next_hop: 100.100.100.100
 BGP.local_pref: 100
2.2.4.0/24 via 192.168.97.101 on em4 [router1 14:01:17 from
100.100.100.100] * (100/?) [AS65001i]
 Type: BGP unicast univ
 BGP.origin: IGP
 BGP.as_path: 65001
 BGP.next_hop: 100.100.100.100
 BGP.local_pref: 250
```

Router R3 – show IP route

On constate ici que la route par défaut est également bien annoncée :

```
@router3:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF, I -
ISIS, B - BGP, > - selected route, * - FIB route
R>* 0.0.0.0/0 [120/2] via 192.168.97.1, eth0, 00:06:15
C>* 1.1.8.0/24 is directly connected, lo
C>* 1.1.9.0/24 is directly connected, lo
S>* 3.3.3.3/32 [1/0] is directly connected, Null0, bh
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.97.0/24 is directly connected, eth0
@router3:~$
```

Dans le cas où ce trafic doit être routé symétriquement - par exemple en cas de NAT - il est nécessaire d'adapter la configuration de BIRD afin d'annoncer le firewall en tant que next-hop. La modification peut se faire dans le filtre « default_ok » qui sert à annoncer la route par défaut à R3 via RIP ainsi qu'à R2 via OSPF :

```
filter default_ok {
  if net = 0.0.0.0/0 then
  {
  dest = RTD_UNREACHABLE; # annonce le firewall comme next-hop pour cette
route
  accept;
  }
}
```

Pour imposer une autre passerelle que le firewall lui-même, il faut utiliser la directive :

```
gw = <ip>;
```

Router R2 – show IP route

```
@router2:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
 I - ISIS, B - BGP, > - selected route, * - FIB route
O>* 0.0.0.0/0 [110/10000] via 192.168.97.1, eth0, 22:26:17
C>* 1.1.1.0/24 is directly connected, lo
C>* 1.1.3.0/24 is directly connected, lo
B>* 2.2.2.0/24 [200/1] via 100.100.100.100 (recursive via 192.168.97.1),
00:02:04
B>* 2.2.4.0/24 [200/1] via 100.100.100.100 (recursive via 192.168.97.1),
00:02:04
C>* 127.0.0.0/8 is directly connected, lo
C>* 192.168.97.0/24 is directly connected, eth0
@router2:~$
```

Router R1 – show IP route

```
@router1:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
I - ISIS, B - BGP, > - selected route, * - FIB route
S>* 0.0.0.0/0 [1/0] via 192.168.97.1, eth0
B>* 1.1.1.0/24 [20/0] via 200.200.200.200 (recursive via 192.168.97.219),
00:00:29
C>* 2.2.2.0/24 is directly connected, lo
C>* 2.2.4.0/24 is directly connected, lo
B>* 3.3.3.3/32 [20/0] via 200.200.200.200 (recursive via 192.168.97.219),
00:00:08
C>* 100.100.100.100/32 is directly connected, lo
C>* 127.0.0.0/8 is directly connected, lo
B>* 172.16.0.0/24 [20/0] via 200.200.200.200 (recursive via
192.168.97.219), 00:00:29
C>* 192.168.97.0/24 is directly connected, eth0
S>* 200.200.200.200/32 [1/0] via 192.168.97.219, eth0
@router1:~$
```

Cas d'une configuration en haute disponibilité (cluster)

En cas d'utilisation du protocole de routage dynamique BGP sur un cluster de firewalls SNS, et pour permettre au voisin BGP de fermer proprement une session BGP en cas de bascule au sein du cluster, il est utile de définir une instance BFD dans la configuration BIRD :

```
protocol bfd mybfdsession {
        neighbor myneighborip;
}
```

Dans cet exemple :

```
protocol bfd mybfdsession {
        neighbor 100.100.100.100;
}
```

Et d'appeler cette instance dans la configuration BGP de la manière suivante :

```
protocol bgp MyBGP {
…
        bfd graceful;
        connect retry time 5;
…
}
```

Migrer une configuration de routage dynamique depuis BIRD v1 vers BIRD v2

La version SNS 4.8.1 introduit le support du moteur de routage dynamique BIRD v2 destiné à remplacer BIRD v1 devenu obsolète.

Lorsque vous mettez à jour en version SNS 4.8.1 ou supérieure un firewall dont la configuration utilisait initialement le routage dynamique en version BIRD v1, cette version BIRD v1 reste active après la mise à jour de firmware.

En effet, le transfert de votre configuration BIRD v1 vers BIRD v2 ne peut pas être réalisé automatiquement, la syntaxe utilisée dans le fichier de configuration du routage dynamique BIRD v2 ayant évolué par rapport à celle de BIRD v1.

L'un des changements importants est que pour BIRD v2, les paramètres de routage dynamique IPv4 et IPv6 sont regroupés dans un unique fichier *bird.conf*, contrairement à BIRD v1 qui utilise deux fichiers distincts : ce même fichier *bird.conf* pour IPv4 et le fichier *bird6.conf* pour IPv6.

Le module **Routage Dynamique** des versions SNS 4.8.1 et supérieures a été conçu afin de pouvoir vous assister dans cette opération de migration.

IMPORTANT
Dans le cas où votre parc de firewalls SNS est géré par un serveur SMC, il n'est plus possible de gérer le routage dynamique de vos firewalls en versions 4.8.1 et supérieures depuis une version de SMC inférieure à la 3.6.

Comprendre le module Routage Dynamique

Placez-vous dans le module **Configuration** > **Réseau** > **Routage Dynamique**.

Ce module comporte 3 onglets permettant d'activer / désactiver l'une ou l'autre des versions de BIRD et de les configurer.

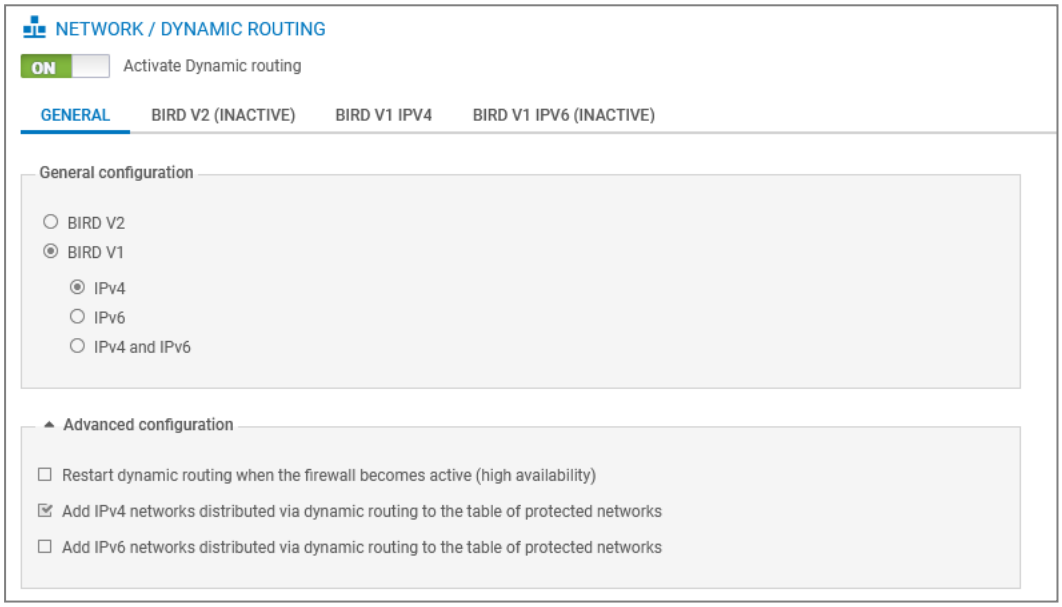
NOTE
Lorsqu'une version de BIRD est désactivée, l'onglet de configuration correspondant affiche la mention "(INACTIVE)".
Exemple : BIRD v2 (INACTIVE).

L'onglet Général

Cet onglet vous permet d'activer / désactiver la version souhaitée du moteur de routage dynamique BIRD.

Après la mise à jour d'un firewall SNS d'une version antérieure à SNS 4.8.1 vers une version SNS 4.8.1 ou supérieure, la configuration est la suivante :

- **BIRD v2** : ce bouton radio est sélectionné par défaut.
- **BIRD v1** : ce bouton radio est sélectionné si le firewall était initialement configuré uniquement en IPv4 s'il possédait une configuration BIRD v1 IPv4 active avant la mise à jour de firmware.
  Les boutons radio suivants ne sont affichés que si le firewall était initialement configuré en IPv4 et IPv6 :
  - **IPv4** : ce bouton radio est sélectionné pour un firewall sur lequel seule une configuration BIRD v1 IPv4 était active avant la mise à jour de firmware.
  - **IPv6** : ce bouton radio est sélectionné pour un firewall sur lequel seule une configuration BIRD v1 IPv6 était active avant la mise à jour de firmware.
  - **IPv4 et IPv6** : ce bouton radio est sélectionné pour un firewall sur lequel des configuration BIRD v1 IPv4 et IPv6 étaient actives avant la mise à jour de firmware.



IMPORTANT
SI vous souhaitez que les routes apprises par BIRD soient ajoutées automatiquement à la table des réseaux protégés et éviter ainsi de générer à tort des alertes d'*antispoofing* concernant ces réseaux, cochez ces cases (suivant votre configuration) :

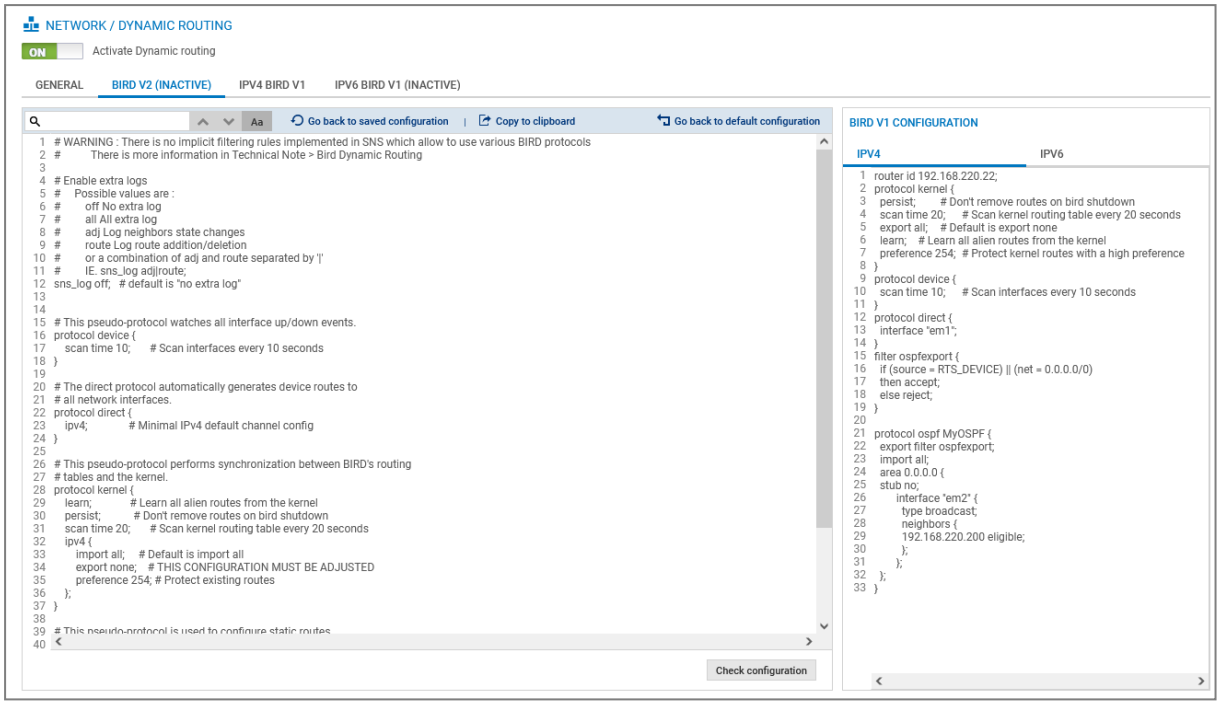
- **Ajouter les réseaux IPv4 distribués par le routage dynamique dans la table des réseaux protégés**.
- **Ajouter les réseaux IPv6 distribués par le routage dynamique dans la table des réseaux protégés**.

L'onglet BIRD v2

Cet onglet affiche :

- Dans la partie gauche de l'écran : une trame de configuration BIRD v2 minimaliste comportant les sections de base obligatoires,
- Dans la partie droite de l'écran : la configuration BIRD v1 d'origine du firewall (IPv4 et / ou IPv6).

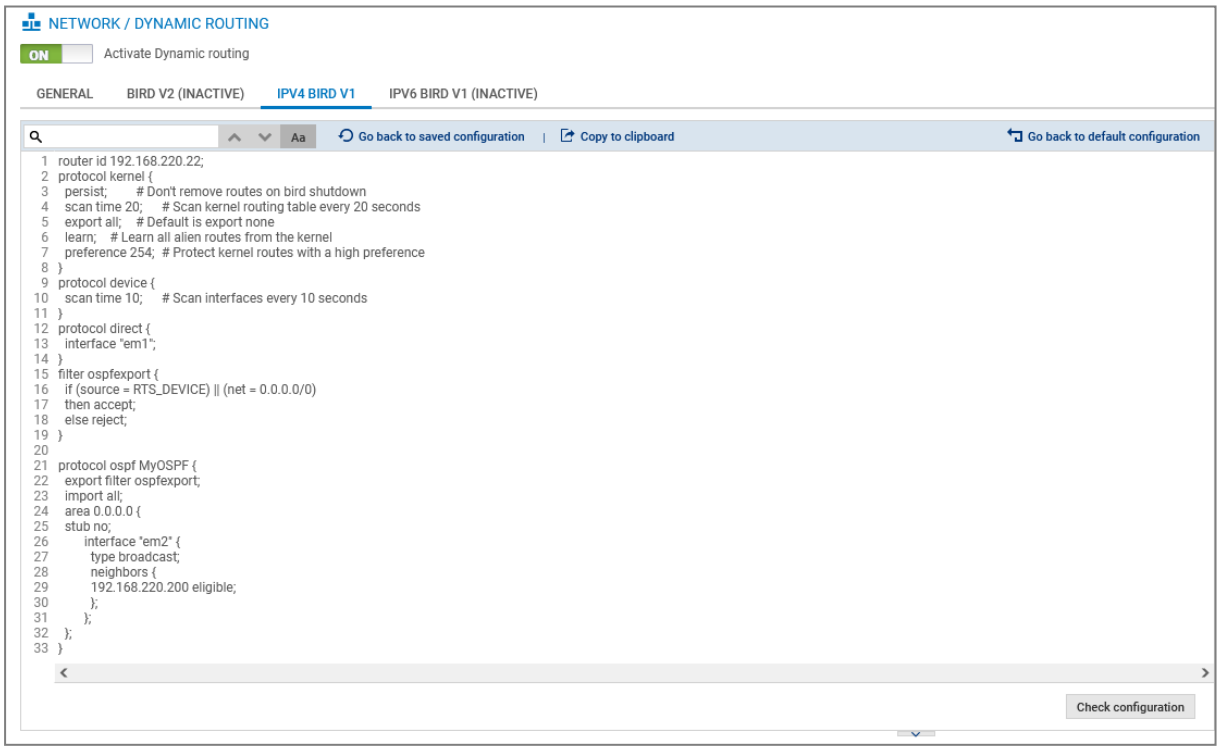
Il vous permet également de modifier la configuration BIRD v2 du firewall et de la valider.



L'onglet BIRD v1 IPv4

Cet onglet affiche la configuration d'origine du firewall pour le routage dynamique IPv4 géré par BIRD v1.

Il vous permet également de la modifier et de la valider.



L'onglet optionnel BIRD v1 IPv6

Cet onglet affiche la configuration d'origine du firewall pour le routage dynamique IPv6 géré par BIRD v1.

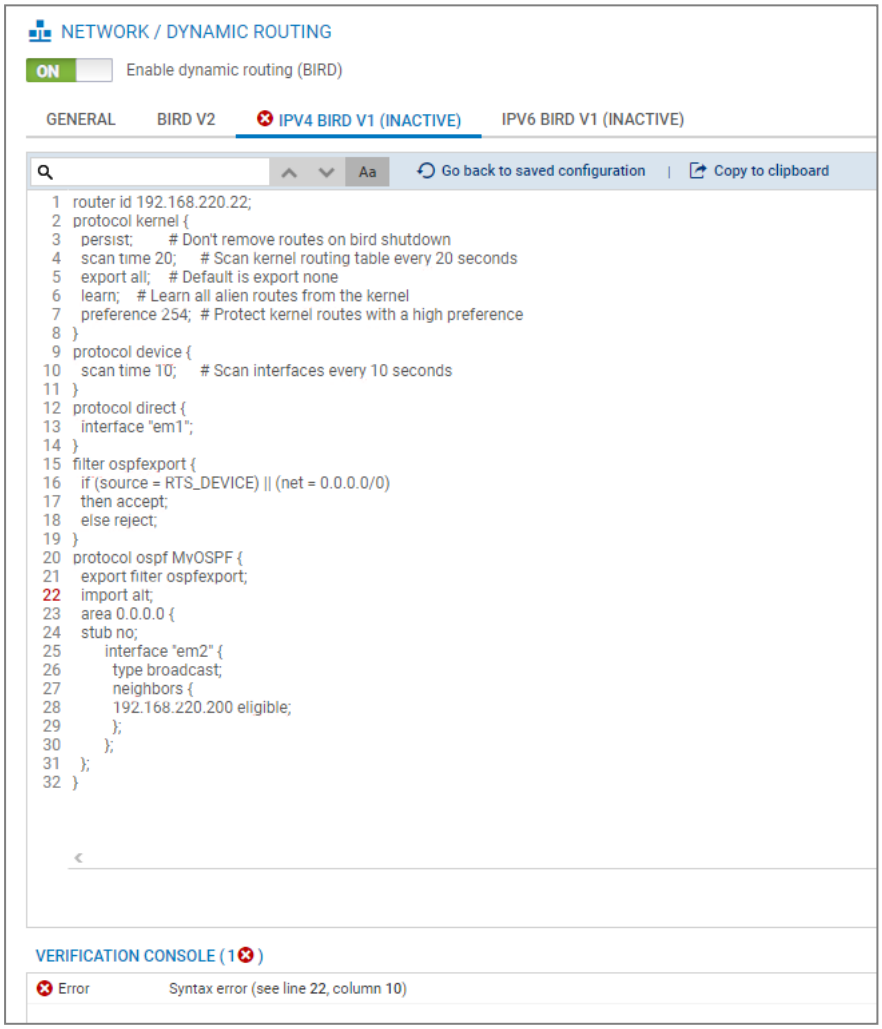
Il vous permet également de la modifier et de la valider.

Sa présentation est identique à celle de l'onglet **BIRD v1 IPv4** / **BIRD v1 IPv4 (INACTIF)**.

La console de contrôle

Lorsque vous cliquez sur le bouton **Vérifier la configuration** depuis l'un des onglets de configuration BIRD présentés ci-dessus, la console de contrôle située en bas de l'écran affiche les éventuelles erreurs de syntaxe rencontrées.

Les erreurs y sont identifiées par leur numéro de ligne et leur numéro de colonne. Les numéros des lignes contenant des erreurs sont également affichés en rouge dans la configuration :



Réaliser la migration BIRD v1 vers BIRD v2

Stormshield vous recommande de suivre la méthode ci-dessous :

Préparer la configuration de BIRD v2

1. Placez-vous dans l'onglet **BIRD v2 (INACTIF)**.
2. En respectant la syntaxe de configuration de BIRD v2, transposez par étapes les informations de votre configuration BIRD v1 (fenêtre de droite) vers la configuration BIRD v2 (fenêtre de gauche). Pour rappel, avec BIRD v2, les paramètres de routage dynamique IPv4 et IPv6 sont regroupés dans un unique fichier *bird.conf*, contrairement à BIRD v1 qui utilise deux fichiers distincts : ce même fichier *bird.conf* pour IPv4 et le fichier *bird6.conf* pour IPv6.

   > **NOTE**
   > Pour vous aider dans cette tâche, consultez les ressources disponibles, notamment le Manuel utilisateur BIRD v2 publié par l'éditeur de BIRD ou les Notes de transition BIRD 1.6 vers BIRD 2.0 (anglais uniquement).

3. Pendant vos modifications, cliquez régulièrement sur le bouton **Vérifier la configuration** après avoir effectué des modifications.
   En bas de l'écran, le vérificateur de cohérence vous affiche alors les erreurs de syntaxe rencontrées dans la configuration BIRD v2.
   Vous ne pouvez pas enregistrer une configuration contenant des erreurs de syntaxe.
4. Lorsqu'une modification de la configuration BIRD v2 est réalisée et est validée (pas d'erreur affichée dans la **Console de contrôle**), enregistrez la configuration en cliquant sur le bouton **Appliquer** puis **Sauvegarder**.
   Cette manipulation crée un point de restauration de la configuration BIRD v2 : si lors d'une modification ultérieure de la configuration BIRD v2 vous ne parvenez pas à résoudre un problème de configuration / syntaxe, cet état de configuration pourra être restauré en cliquant sur le bouton **Revenir à la version enregistrée**.
5. Lorsque votre migration est complète et que vous avez enregistré votre configuration BIRD v2, vous pouvez activer BIRD v2 afin de contrôler le bon fonctionnement du routage dynamique.

Contrôler le fonctionnement du routage dynamique

Après avoir activé BIRD v2, si vous détectez que le fonctionnement du routage dynamique n'est pas conforme à vos attentes :

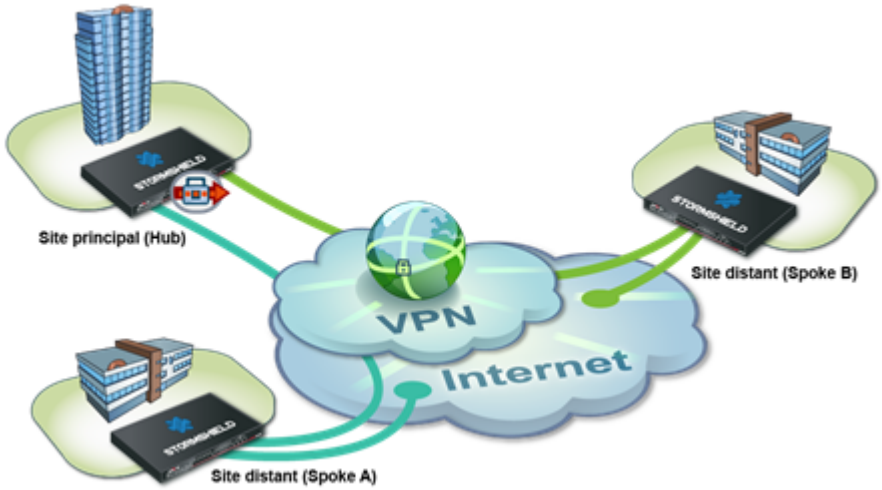
1. Depuis l'onglet **Général**, désactivez BIRD v2 et réactivez BIRD v1 afin de revenir à la situation avant migration de BIRD.
   Vous pouvez alors corriger votre configuration BIRD v2 tout en ayant le routage dynamique BIRD v1 actif.
2. Réactivez BIRD v2 une fois la configuration corrigée.

Ces opérations peuvent être réalisées autant de fois que nécessaire.

Annexe A : Tunnels VPN Hub and Spoke routés via BGP

Voici un exemple de routage dynamique BGP dans le cadre d'un réseau VPN en étoile de type Hub and Spoke.

Configuration des tunnels



Pour le paramétrage de la politique IPsec Hub and Spoke, consultez le **cas n° 1 : trafic interne via les tunnels IPsec** de la Note Technique VPN IPsec - Configuration Hub and Spoke.

Dans notre cas, les différences de paramétrage par rapport à cette procédure consistent à configurer les extrémités de trafic au moyen d'interfaces virtuelles, au lieu de réseaux distants dans la politique IPsec :

Site principal

TunnelA

Réseau local :	Interface ipsec1 (172.16.0.1)
Correspondant :	Site_SpokeA
Réseau distant :	Remote_tunnelA (172.16.0.2)

TunnelB

Réseau local :	Interface ipsec2 (172.16.0.5)
Correspondant :	Site_SpokeB
Réseau distant :	Remote_tunnelB (172.16.0.6)

Spoke A

Réseau local :	Interface ipsec1 (172.16.0.2)
Correspondant :	Site_FW_Hub
Réseau distant :	Remote_tunnelA (172.16.0.1)

Spoke B

Réseau local :	Interface ipsec1 (172.16.0.6)
Correspondant :	Site_FW_Hub
Réseau distant :	Remote_tunnelB (172.16.0.5)

Configuration BGP du site principal (Hub)

```
protocol direct {
}

protocol kernel {
       learn;# Learn all alien routes from the kernel
       persist;# Don't remove routes on bird shutdown
       scan time 20;# Scan kernel routing table every 20 seconds
       ipv4 {
            import all;# Default is import all
            export all;# Default is export none
            preference 254;# Protect existing routes
       };
}

# This pseudo-protocol watches all interface up/down events.
       protocol device {
       scan time 10;# Scan interfaces every 10 seconds
}

filter f_import {
       if source = RTS_BGP then
       accept;
       else
       reject;
}

filter f_export {
# local shared networks and BGP routes
       if( (net =  192.168.0.0/24) || (source  = RTS_BGP) ) then
       accept;
       else
       reject;
}

router id <ip_pub_hub>;

template bgp star {
       local as 65000;
       ipv4 {
           import filter f_import;
           export filter f_export;
           next hop self;
       };
       hold time 5;
       multihop;
       rr client;
}

protocol bgp router_spokeA from star {
       neighbor 172.16.0.2 as 65000;
       source address 172.16.0.1;
}
```

```
     protocol bgp router_spokeB from star {
            neighbor 172.16.0.6 as 65000;
            source address 172.16.0.5;
     }
```

Configuration BGP du site satellite Spoke A

```
protocol direct {
}

protocol kernel {
       learn;# Learn all alien routes from the kernel
       persist;# Don't remove routes on bird shutdown
       scan time 20;# Scan kernel routing table every 20 seconds
       ipv4 {
            import all;# Default is import all
            export all;# Default is export none
            preference 254;# Protect existing routes
       };
}

protocol device {
       scan time 10;# Scan interfaces every 10 seconds
}

filter filter_export_net {
       if(net = 192.168.1.0/24) then {
       accept;
       }
       else reject;
}

router id  <ip_pub_spokeA>;

protocol bgp router_tunnel1 {
       local as 65000;
       neighbor 172.16.0.1 as 65000;
       hold time 5;
       multihop;
       ipv4{
            import all;
            export filter filter_export_net;
       };
source address 172.16.0.2;
}
```

Configuration BGP du site satellite Spoke B

```
protocol direct {
}

protocol kernel {
       learn;# Learn all alien routes from the kernel
       persist;# Don't remove routes on bird shutdown
       scan time 20;# Scan kernel routing table every 20 seconds
       ipv4 {
            import all;# Default is import all
            export all;# Default is export none
            preference 254;# Protect existing routes
       };
}

protocol device {
       scan time 10;# Scan interfaces every 10 seconds
}

filter filter_export_net {
       if(net = 192.168.2.0/24) then {
       accept;
       }
       else reject;
}

router id  <ip_pub_spokeB>;

protocol bgp router_tunnel2 {
       local as 65000;
       neighbor 172.16.0.5 as 65000;
       hold time 5;
       multihop;
       ipv4{
            import all;
            export filter filter_export_net;
       };
       source address 172.16.0.6;
}
```

Vérification des tables de routage

Table de routage sur le site principal (Hub) :

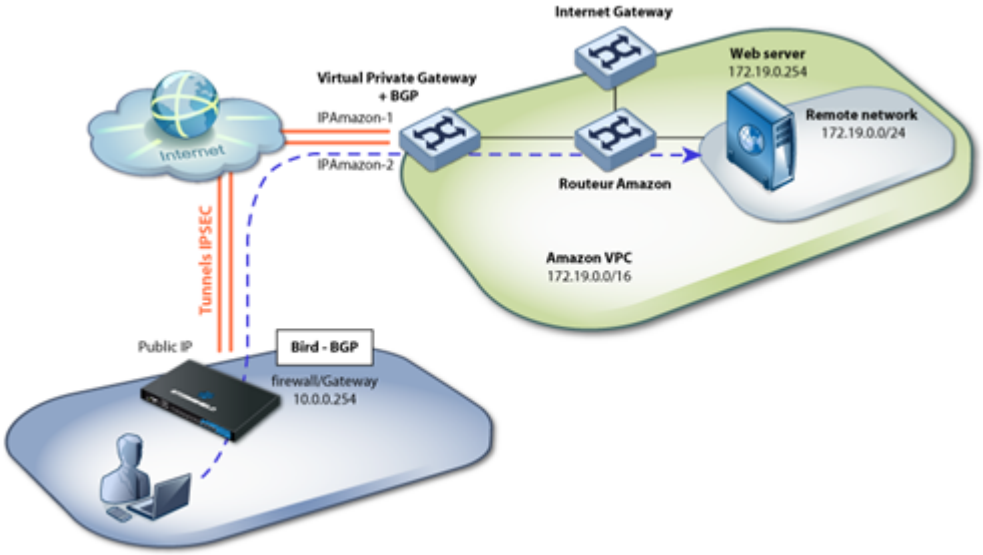
```
bird> show route
0.0.0.0/0              via 10.60.0.254 on em0 [kernel1 10:16] * (254)
10.60.3.127/32         dev lo0 [kernel1 10:16] * (254)
192.168.0.0/24         dev em1 [direct1 10:16] * (240)
192.168.1.0/24         dev em2 [direct1 10:16] * (240)
192.168.1.0/24         via 172.16.0.2 on enc1 [router_tunnelA 10:22]*(100/0)
[AS65001i]
192.168.2.0/24         via 172.16.0.6 on enc1 [router_tunnelB 10:21]*(100/0)
[AS65002i]
192.168.0.254/32       dev lo0 [kernel1 10:16] * (254)
192.168.1.254/32       dev lo0 [kernel1 10:16] * (254)
172.16.0.0/30          dev lo1 [direct1 10:16] * (240)
10.60.0.0/16           dev em0 [direct1 10:16] * (240)
172.16.0.4/30          dev lo2 [direct1 10:16] * (240)
```

Table de routage sur spokeA :

```
bird> show route
0.0.0.0/0       via 10.60.0.254 on em0 [kernel1 13:32] * (254)
192.168.0.0/24 via 172.16.0.1 on enc1 [router_tunnelA 13:32] * (100/0) [i]
192.168.2.0/24 via 172.16.0.1 on enc1 [router_tunnelA 13:32] * (100/0) [i]
192.168.1.0/24 dev em1 [direct1 13:32] * (240)
172.16.0.0/30  dev lo1 [direct1 13:32] * (240)
10.60.3.128/32 dev lo0 [kernel1 13:32] * (254)
10.60.0.0/16   dev em0 [direct1 13:32] * (240)
```

Annexe B : Connectivité Amazon VPC

Le but est de relier un réseau local à un VPC Amazon (Virtual Private Cloud). Pour cela, Amazon propose la création de deux tunnels routés entre le firewall local et le Cloud Amazon et de router ce trafic via BGP.



Configuration Amazon

Suivez les étapes ci-dessous :

1. Créez un VPC Amazon.
2. Créez un sous réseau dans ce VPC.
3. Configurez le routage dans ce VPC.
4. Créez une connexion VPN dynamique vers le firewall via l'objet Amazon Virtual Private Gateway.
5. Créez les ACLs pour autoriser le trafic local vers le serveur Web.
6. Routage : activez la propagation des routes à la table de routage du VPC.

Extrait de l'aide de configuration fournie par Amazon lors de la configuration du service :

```
The Customer Gateway inside IP address should be configured on your tunnel
interface.

Outside IP Addresses:
      • Customer Gateway: IP publique Firewall/Gateway,
      • Virtual Private Gateway: IPAmazon-1.

Inside IP Addresses:
      • Customer Gateway : 169.254.254.66/30,
      • Virtual Private Gateway : 169.254.254.65/30.

Configure your tunnel to fragment at the optimal size:
      • Tunnel interface MTU : 1436 bytes.

#4: Border Gateway Protocol (BGP) Configuration:
The Border Gateway Protocol (BGPv4) is used within the tunnel, between the
```

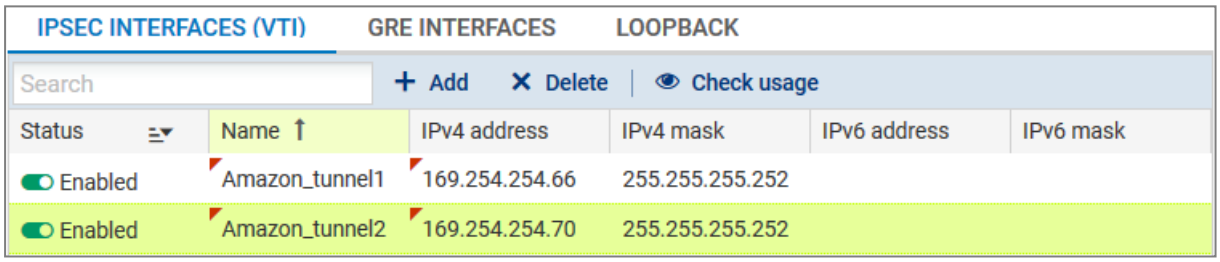
```
inside IP addresses, to exchange routes from the VPC to your home network.
Each BGP router has an Autonomous System Number (ASN). Your ASN was
provided to AWS when the Customer Gateway was created.

BGP Configuration Options:
```

- Customer Gateway ASN : 65000,
- Virtual Private Gateway ASN : 9059,
- Neighbor IP Address : 169.254.254.65,
- Neighbor Hold Time: 30.

```
Configure BGP to announce routes to the Virtual Private Gateway. The
gateway will announce prefixes to your customer gateway based upon the
prefix you assigned to the VPC at creation time.
```

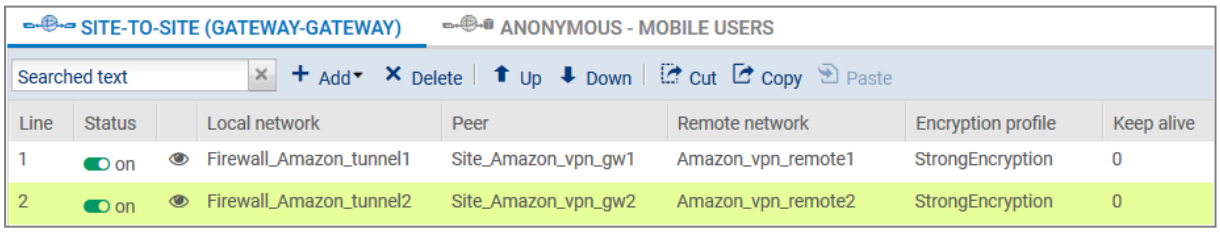
Configuration des tunnels

Dans le module **Configuration** > **Réseau** > **Interface virtuelles**, l'onglet *Interfaces IPsec* vous permet de définir les interfaces concernées :



Dans le module **Configuration** > **VPN** > **VPN IPsec**, onglet *Site à site (gateway-gateway)*, vous pouvez définir les tunnels ci-dessous, à l'aide des objets suivants :

- **Site_Amazon_vpn_gw1** : IPAmazon-1,
- **Site_Amazon_vpn_gw2** : IPAmazon-2,
- **Amazon_vpn_remote1** : 169.254.254.65,
- **Amazon_vpn_remote2** : 169.254.254.69.



Configuration BGP

On choisit d'exporter uniquement le réseau 10.0.1.0/24

```
filter filter_net_in {
       if net = 10.0.1.0/24 then {
       accept;
       }
       else reject;
}

protocol bgp router1 {
```

```
        local as 65000;
        neighbor 169.254.254.65 as 9059;
        source address 169.254.254.66;
        hold time 30;
        multihop;
        ipv4 {
                import all;
                export filter filter_net_in;
        };
}

protocol bgp router2 {
        local as 65000;
        neighbor 169.254.254.69 as 9059;
        source address 169.254.254.70;
        hold time 30;
        multihop;
        ipv4 {
                export filter filter_net_in;
                import all;
        };
}
```

Pour aller plus loin

Des informations complémentaires et réponses à vos éventuelles questions sur le routage dynamique BIRD sont disponibles dans la base de connaissances Stormshield (authentification nécessaire).

documentation@stormshield.eu