TECHNICAL NOTE
STORMSHIELD NETWORK SECURITY

BIRD V2 DYNAMIC ROUTING

Product concerned: SNS 4.8.1 and higher versions

Document last updated: September 25, 2024

Reference: sns-en-BIRD_v2_dynamic_routing_technical_note

Table of contents

Change log

Date	Description
September 25, 2024	New document

Introduction

As of SNS version 4.8.1, the BIRD v2 dynamic routing engine will be supported, and replaces BIRD v1, which has become obsolete.

The aim of this document is to guide SNS firewall administrators through the configuration and operation of the built-in BIRD v2 dynamic routing module.

To migrate your BIRD v1 configuration to BIRD v2, refer to the section Migrating a dynamic routing configuration from BIRD v1 to BIRD v2

The first part of the document describes the configuration environment, and the modes of interaction with the routing engine. Next, a simple typical configuration will be presented for the BGP, RIP and OSPF routing protocols.
These examples provide the opportunity to discover the configuration structure of the protocols as well as peripheral objects, filtering and status displays. The last section will focus on a more complex configuration.

Do note that BIRD offers various configuration options, notably inter-process route exchanges, route filtering, and the pseudo-virtualization of routing instances. These advanced options, which are specific to BIRD, will not be covered in this document. Likewise, the use of BGP ROAs will not be explained, but do note that they are supported by BIRD.

These components, as well as the detailed syntax used in the dynamic routing configuration, are provided in the BIRD v2 user guide published on the BIRD vendor website.

Restrictions

IMPORTANT
If your SNS firewall pool is managed by an SMC server, you will need SMC in version 3.6 or higher in order to manage dynamic routing on your firewalls in 4.8.1 and higher versions.

Understanding the Dynamic routing module

Dynamic routing can be configured in the **General** tab of the **Dynamic routing** module in the web administration interface.

In the **BIRD v2**, **IPv4 BIRD v1** and **IPv6 BIRD v1** tabs (if IPv6 support has been enabled), BIRD configuration files can be edited.

Do note that the graphical interface editor does not allow you to access interactive modes, in which dynamic routing can be monitored (operational tests of new configurations through a temporary configuration and status preview).

The table below maps each BIRD version to its configuration tab in the web administration interface, its configuration file, and its interactive binary file in console mode:

BIRD version	Configuration tab	Configuration file	Interactive binary file
BIRD v2 - IPv4 and IPv6	BIRD v2	bird.conf	birdc
BIRD v1 - IPv4	IPv4 BIRD v1	bird4.conf	birdc4
BIRD v1 - IPv6	IPv6 BIRD v1	bird6.conf	birdc6

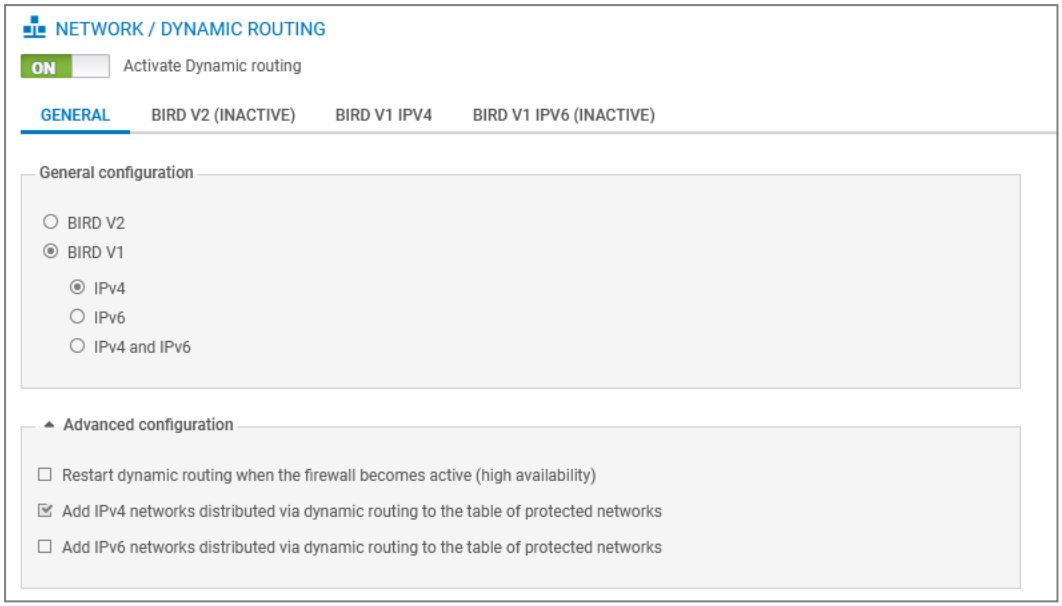
NOTE
When one version of BIRD is disabled, the corresponding configuration tab will show the suffix " (INACTIVE)".
Example: BIRD v2 (INACTIVE).

General tab

This option allows you to enable/disable the desired version of the BIRD dynamic routing engine.

After SNS firewalls in a version lower than SNS 4.8.1 are updated to SNS version 4.8.1 or higher, the configuration will be as follows:

- **BIRD v2**: this radio button is selected by default.
- **BIRD v1**: this radio button will be selected if the firewall was initially configured only in IPv4, and if its IPv4 BIRD v1 configuration was active prior to the firmware update.
  The following radio buttons will appear only if the firewall was initially configured in IPv4 and IPv6:
  - **IPv4**: this radio button is selected for firewalls on which only an IPv4 BIRD v1 configuration was active prior to the firmware update.
  - **IPv6**: this radio button is selected for firewalls on which only an IPv6 BIRD v1 configuration was active prior to the firmware update.
  - **IPv4 and IPv6**: this radio button is selected for firewalls on which IPv4 and IPv6 BIRD v1 configurations were active prior to the firmware update.



IMPORTANT

If you want the routes that were learned by BIRD to be automatically added to the table of protected networks, thereby preventing these networks from wrongly raising antispoofing alerts, select these checkboxes (depending on your configuration):

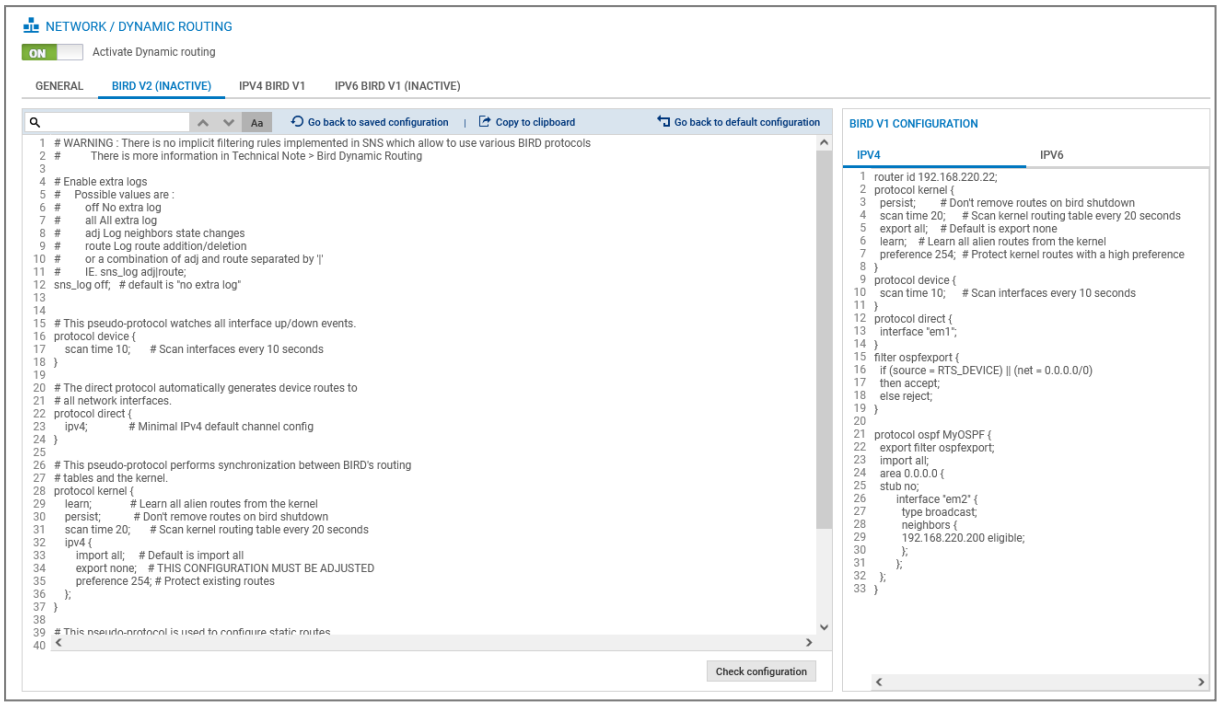
- **Add IPv4 networks distributed via dynamic routing to the table of protected networks**.
- **Add IPv6 networks distributed via dynamic routing to the table of protected networks**.

BIRD v2 tab

This tab shows:

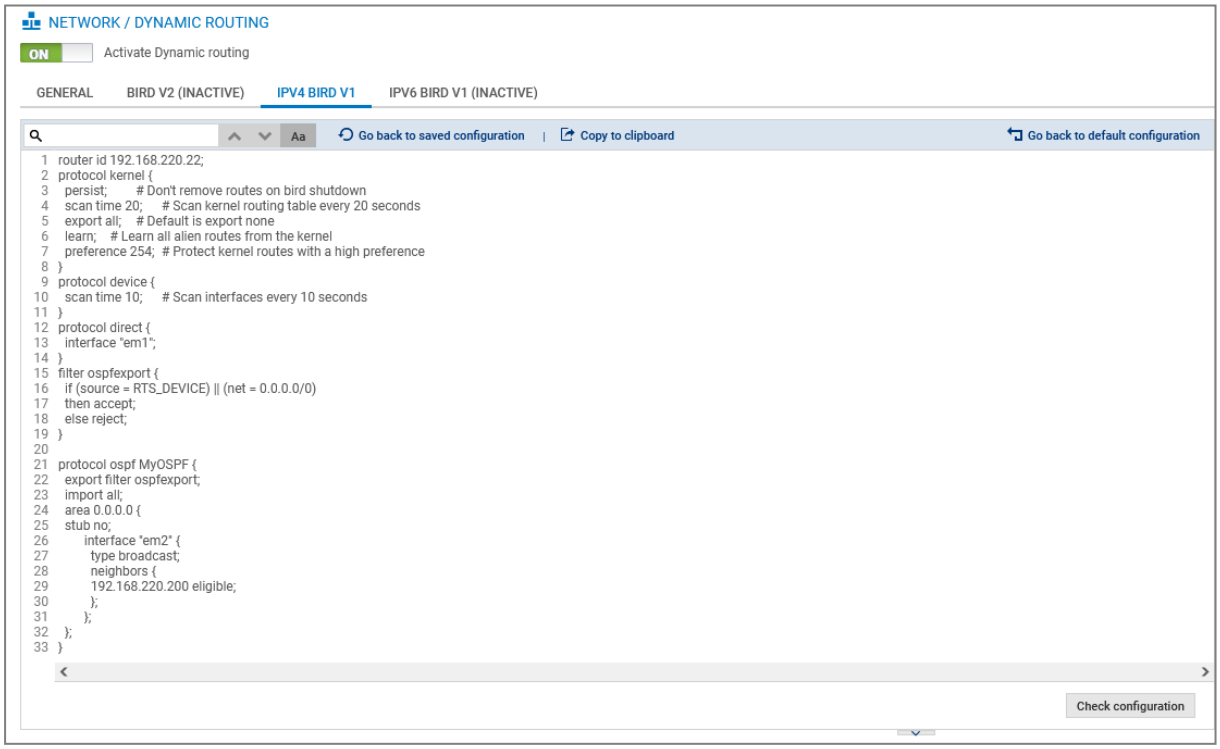
- On the left side of the screen: a minimalist BIRD v2 configuration frame containing the basic mandatory sections,
- On the right side of the screen: the firewall's original BIRD v1 configuration (IPv4 and/or IPv6).

This section also allows you to modify the firewall's BIRD v2 configuration and validate it.



IPv4 BIRD v1 tab

This tab shows the original configuration on the firewall for the IPv4 dynamic routing managed by BIRD v1. This section also allows you to edit and validate the configuration.



Optional IPv6 BIRD v1 tab

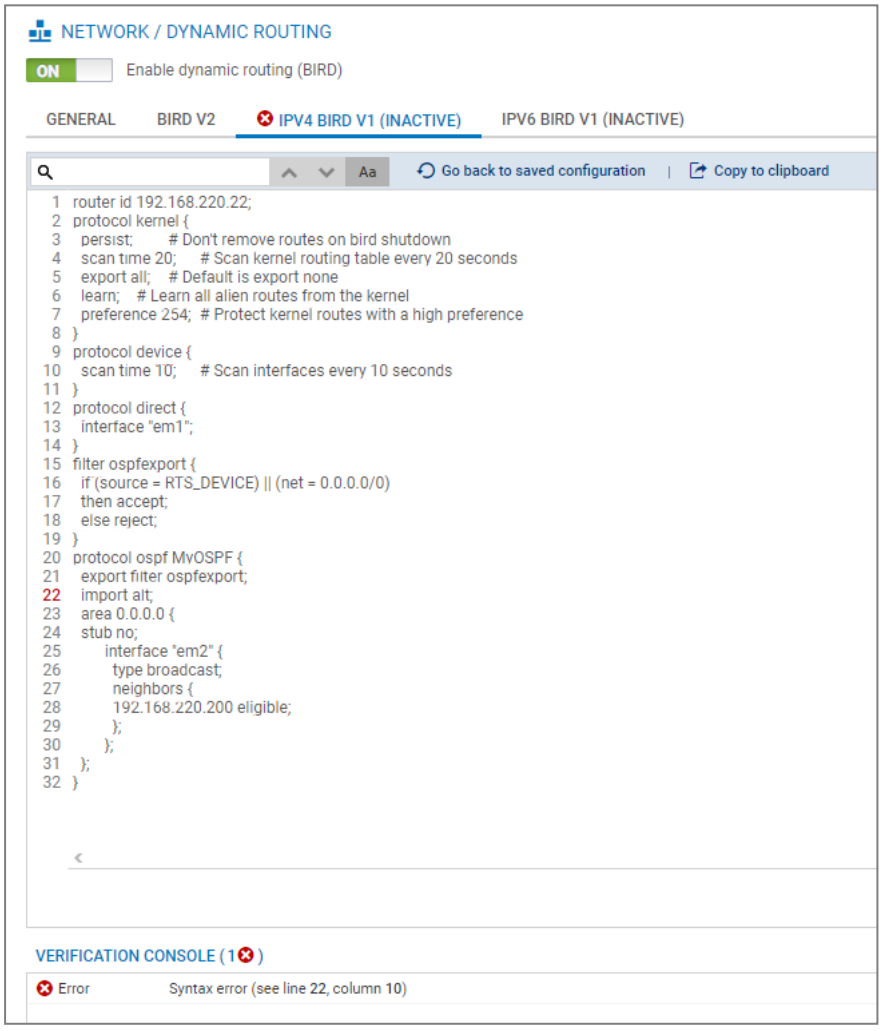
This tab shows the original configuration on the firewall for the IPv6 dynamic routing managed by BIRD v1. This section also allows you to edit and validate the configuration.

It looks exactly like the **IPv4 BIRD v1/IPv4 BIRD v1 (INACTIVE)** tab.

Verification console

When you click on the **Check configuration** button in one of the BIRD configuration tabs shown below, the verification console located at the bottom of the screen shows the syntax errors encountered, if any.

Errors are identified in the console by their line numbers and column numbers. Line numbers that contain errors are also highlighted in red in the configuration:



Navigating the BIRD/Stormshield Network environment

In factory configuration, the BIRD routing mode is not enabled.

Stormshield Network firewall routing can coexist with BIRD dynamic routing. For example, the internal zone can be managed with a dynamic routing protocol, while the external zone is managed with the firewall's routing features (static routing, gateways, policy-based routing (PBR), or router objects).

Refer to the section Interactions with Stormshield Network routing.

Unlike BIRD v1, which used two separate configuration files, BIRD v2 dynamic routing for IPv4 and IPv6 is configured in a single file: **/usr/Firewall/ConfigFiles/Bird/bird.conf**.

Starting BIRD v2 routing from the web administration interface

To enable and start BIRD v2 routing:

1. Go to **Configuration > Network > Dynamic routing** > **General** tab,
2. Switch the Enable dynamic routing cursor to **ON**,
3. In the **General configuration** section, select the **BIRD v2** radio button.

Monitoring BIRD v2 dynamic routing in interactive mode

BIRD offers an interactive mode: ***birdc*** for BIRD Client.

In this interactive mode, you can view BIRD statuses, test the operation of a new configuration with the possibility of backtracking, and create temporary configurations.

However, this interactive mode does not allow you to make permanent changes to the BIRD configuration file.

In the firewall's console, run this mode by calling up **birdc** to monitor dynamic routing.

The first detail that is shown is the BIRD version:

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

"Show" commands

The “?” character makes it possible to to display the list of available options:

```
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
```

Example:

Show all 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)
```

Example:

Show routes by protocol instance. The instance in this case is **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>
```

In ***birdc***, most of the commands are the same across protocols. So for example, the routes announced to a BGP neighbor are viewed by a command that relies on the export filter (filter named *ospfexport* in this example):

```
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

Show commands provide many details on instances. With these commands, issues can be diagnosed, whether they are due to a wrong configuration, network issue or other cause.

```
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
```

To enable reception of system messages on the console, enter the command "echo all", then "echo off" to stop such 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
```

Debug events are viewed globally or for example, by protocol instance. The use of *debug* commands effectively complements status visualization commands.

```
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
```

Temporary test of new configurations

Here, we would like to test a new configuration **bird_conf_to_test.conf**. To do so, enable BIRD by using a **bird.conf** configuration that has been validated, then run the interactive **birdc** mode from the firewall's console.

To check the syntax of the file without applying it:

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

Next, apply this configuration temporarily for 60 seconds by using the command:

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

The new configuration will be applied. If the firewall can no longer be reached, or if the administrator has not given any confirmation, the previous configuration will be automatically applied once more after 60 seconds.

If the new configuration is considered valid, it can be confirmed using:

```
bird> configure confirm
```

If the new configuration has not been validated, and the firewall is still reachable, you can immediately backtrack by using:

```
bird> configure undo
```

Configuring dynamic routing

BIRD v2 can be configured in **Configuration** > **Network** > **Dynamic routing** > **BIRD v2 (INACTIVE)** tab.

In any implementation, the following lines must at least be configured in order to define a basic environment for cooperation with the system.

```
# 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
}
```

We will not cover each configuration line in detail here. If you require more exhaustive explanations, refer to the online BIRD documentation at:

http://bird.network.cz/?get_doc&f=bird.html.

The most important concepts to note are protocol instances and filters.

A protocol instance can either be BGP, RIP or OSPF, and it defines an appropriate configuration. Several instances can be defined when necessary for the same protocol.

Each protocol instance is connected to an internal BIRD routing table. This connection is monitored by two filters that can accept, reject or modify routes.

The export filter monitors routes that are sent from the internal BIRD routing table to the protocol. The import filter performs the same operation in the opposite direction.

IMPORTANT
Precision is key during the implementation of route filtering.
The use of full route imports or exports (e.g., *import all;*) between protocol instances may produce disastrous results.

Understanding syntax rules

- Any text on the line placed after a # is a comment,
- Any text between /* and */ is a comment,
- Blocks of several options are placed between curly brackets {},
- Every option ends with a semi-colon ;,
- The configuration is case sensitive.

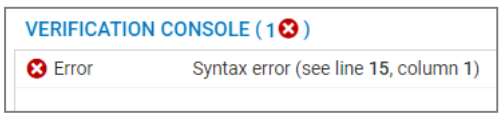
Verifying a configuration

The example described below shows how to verify a configuration that contains two syntax errors.

The following configuration is entered in the configuration window:

```
# 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
}
```

Click on **Check configuration**. The verification console located at the bottom of the screen will indicate the first error that it encounters.

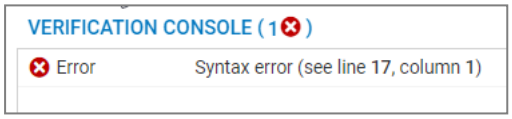


Line number 15 is also highlighted in red in the configuration.

```
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
```

Explanation: if there is no curly bracket to close a block, the error will mention the **first line of the following block**, which is a line that does not match the command allowed in the unclosed block. The "**}**" character therefore needs to be inserted at the end of the previous line.

Once the first error has been fixed, click again on **Check configuration** to show the following syntax error in the verification console (with the line number corresponding to the error highlighted in red in the 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
```

Explanation: the "**;**" character has to be inserted at the end of the previous line (line 16 in this example).

Interactions with Stormshield Network routing

In the default configuration provided on Stormshield Network firewalls, the firewall's routing module takes priority over dynamic routing (maximum preference value of 254).

WARNING
When the firewall's routes are reconfigured, they are temporarily erased, so BIRD can configure its own routes. The firewall's routing has to be protected using an export filter on the *kernel* pseudo-protocol.

The following is an example of a filter that protects the default route and static route 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
       };
}
```

Dynamic routing with priority over Stormshield Network routing

In the BIRD dynamic routing table, if you want dynamic routing to take priority over Stormshield Network routing, the routes obtained by dynamic routing (BGP, OSPF or RIP protocol) must have a higher preference value than the routes obtained by the system (*kernel* pseudo-protocol). However, this will have no impact on the firewall's routing table itself (which can be shown using the command `netstat -r`).

The preference value of the *kernel* must therefore be lowered, for example to 1:

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

Routing the firewall's interfaces

If the firewall's interfaces have been configured with different sub-networks, and you wish to forward interface sub-networks via BIRD, use the *direct* pseudo-protocol.

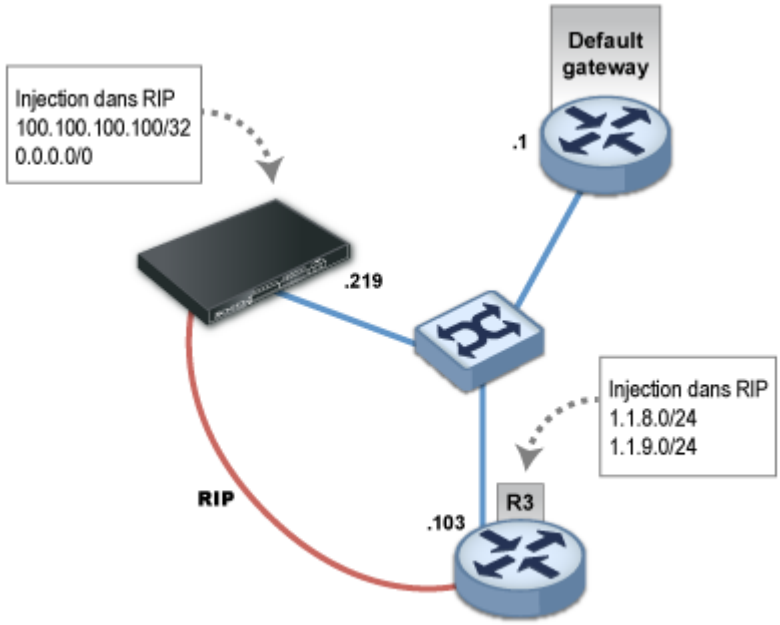
By default, all interfaces will be taken into account. You can restrict the set of interfaces that are taken into account by using the *interface* attribute.

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

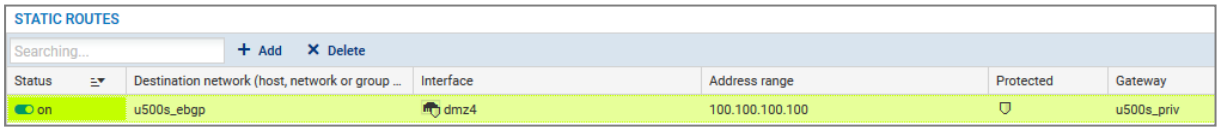
Understanding simple configurations

RIP

The version supported is RIP v2.
The following is the "RIP_simple" configuration.



Configure a default route and a static route to 100.100.100.100/32:



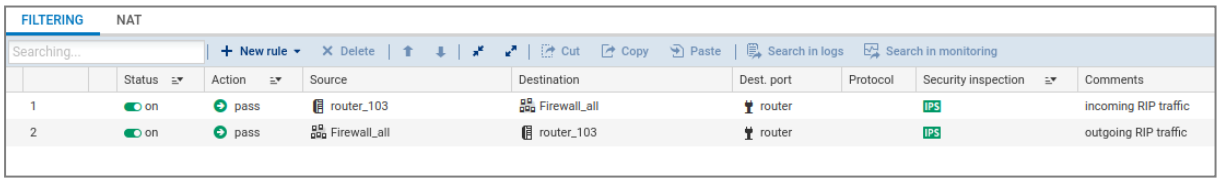
Configure RIP v2 by specifying "multicast" as the RIP mode associated with the interface "em0".

```
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;
}
```

Allowing the RIP protocol in filter policies

Filter rules are required in order to allow RIP routing traffic to and from the firewall:



Checking the proper operation of RIP dynamic routing

Show the status of the protocol instance:

```
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
```

Show learned routes:

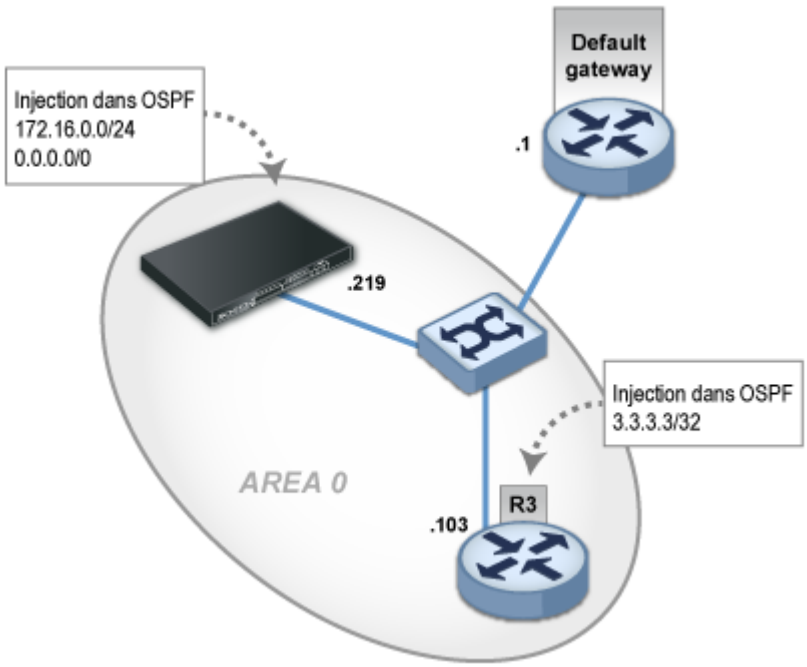
```
bird> show route primary protocol MyRIP
192.168.97.0/24 via 10.200.45.250 on eth0 [MyRIP 10:29:19] ! (120/2)
1.1.9.0/24 via 10.200.45.250 on eth0 [MyRIP 10:29:19] * (120/2)
1.1.8.0/24 via 10.200.45.250 on eth0 [MyRIP 10:29:19] * (120/2)
```

The following are the routes that were received by the neighbor. You will notice that the default route has been received. Mainstream routers will ordinarily not allow this route to be exported. In this case, it has to be explicitly filtered.

```
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

The supported versions are OSPF v2 for IPv4, and OSPF v3 for IPv6.
The following is the "OSPF_simple" configuration.



It consists of deploying an area 0 over a LAN in which any neighbors are explicitly designated. All routes are imported from OSPF. The route of the subnet that is directly connected to the interface em3 (172.16.0.0/24), as well as the default route, are then redistributed in OSPF.

```
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;
        };
}
```

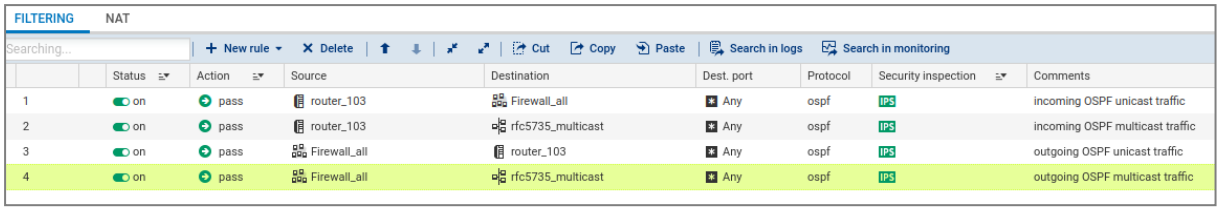
NOTE
You are advised to set the value of the parameter `priority 0` in the *interface* section of the OSPF node configuration, in order to disable the firewall's participation in elections for Designated Router/Backup Designated Router roles.

Allowing the OSPF protocol in filter policies

Filter rules are required in order to allow OSPF routing traffic to and from the firewall.

In the example of a filter policy below, the object *router_103* represents the IP address (192.168.97.103) of the OSPF neighbor that was explicitly declared in the firewall configuration.

IMPORTANT
To ensure that OSPF operates correctly, you must allow OSPF unicast traffic in addition to multicast traffic, as shown in this example of a filter policy.



Checking the proper operation of OSPF dynamic routing

The command below indicates that the neighborhood has been set up ("full" status)
The neighbor is declared as the "Designated Router" ("dr" status):

```
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
```

Received routes:

```
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]
```

The OSPF topology can be shown:

```
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
```

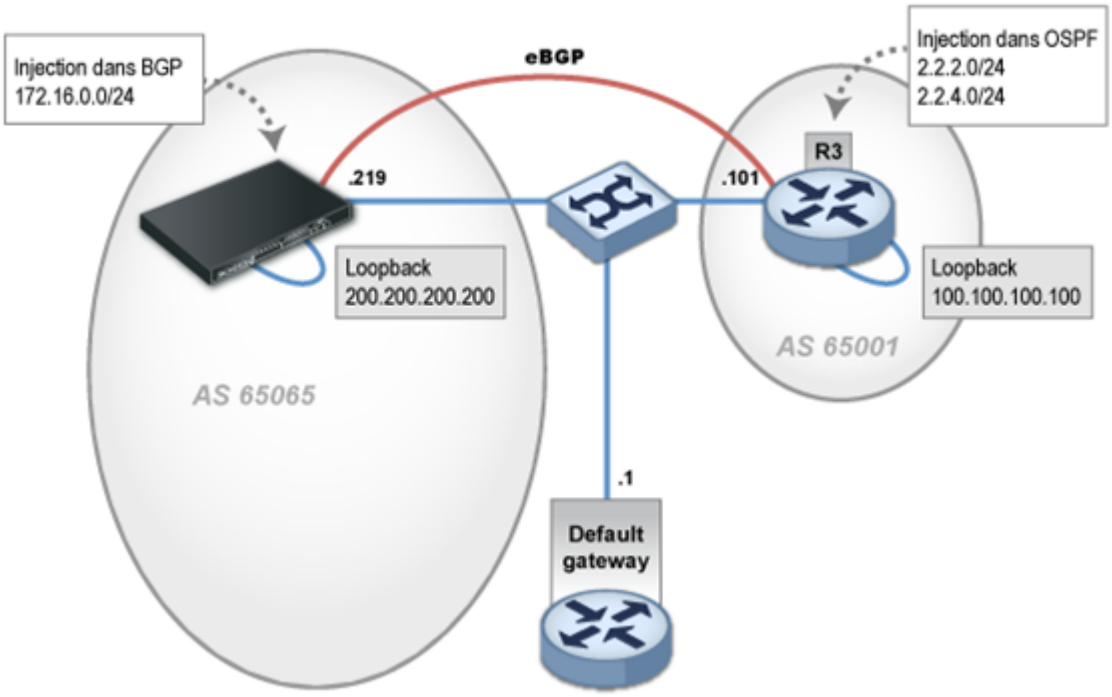
As well as the LSA database:

```
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
Do note that the LSA type is shown on the left although it is generally used as the horizontal delimiter in conventional displays.

BGP

The supported version is BGP v4 for IPv4 and IPv6.
The following is the "BGP simple" configuration:



The "BGP_simple" configuration is implemented as follows:

```
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;
}
```

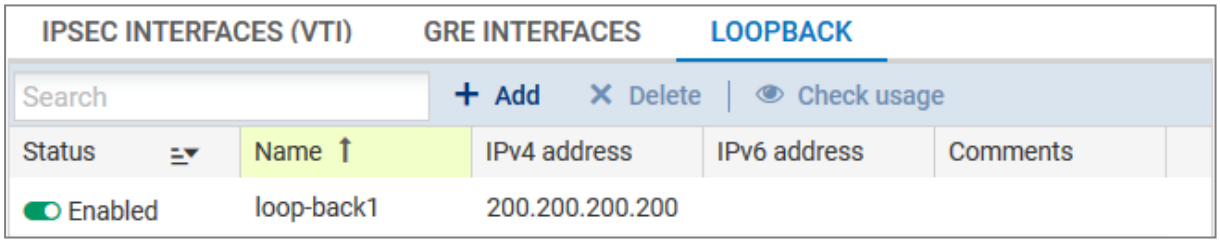
Explanations

Unlike most mainstream routers, the local AS has to be specified for each BGP instance.

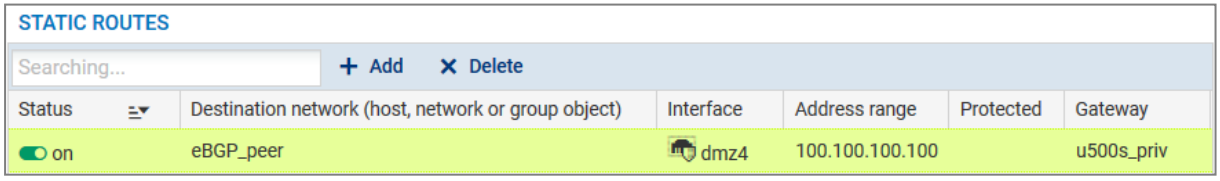
By following best practices, this eBGP session has to be set up between loopback interfaces, instead of physical interfaces. The IP address of the local loopback in question therefore has to be configured (200.200.200.200/32), by specifying this address as the source, and a static route to the neighbor's loopback.

Loopback virtual interfaces

Loopback interfaces can be configured in the web administration interface, in **Configuration** > **Network** > **Virtual interfaces, Loopback** tab:



We recommend declaring the static route to the remote loopback on the firewall outside the BIRD configuration, in **Configuration** > **Network** > **Routing**, *Static routes* tab, to prevent BGP traffic from being blocked by “IP address spoofing” alarms:



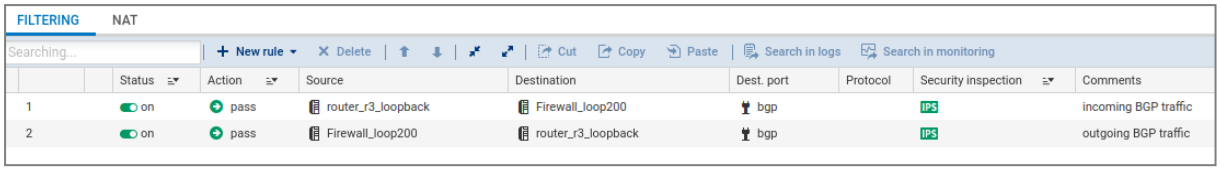
Once again, we will select only the sub-network 172.16.0.0/24, which is directly connected to the interface *em3* as the route to be announced to our neighbors.

In this case, we have defined an anonymous export filter, directly in the "export” instruction, with "where” as the keyword. This export filter selects the routes that have RTS_DEVICE as their source, i.e., routes that were obtained by the direct pseudo-protocol.

The *hold-time* value has been set to 180s, a standard mainstream value. By default, BIRD implements 240s. The *keepalive* value (calculated as 1/3 of the *hold-time*) does not need to be specified, but we are mentioning it explicitly for clarity. The same goes for the default *local-preference* value.

Allowing the BGP protocol in filter policies

Filter rules are required in order to allow BGP routing traffic to and from the firewall.



Checking the proper operation of BGP dynamic routing

The “show protocols” command below confirms that the session is functioning.

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

Routes have been received from the neighbor:

```
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?]
```

The BGP neighbor receives the route that has been announced and released by the filter. As for route 1.1.1.1/32, it has been blocked.

Authentication

TCP-MD5 authentication can be set up between BGP routers in a BIRD configuration.

With this method, BGP sessions can be protected through the authentication of frames in the TCP header, in line with RFC2385.

This involves adding the *"password"* directive in the BGP router configuration in the **/usr/Firewall/ConfigFiles/Bird/bird.conf** files (dynamic routing of IPv4 and IPv6 packets). The “source address" directive must also be added, by specifying the IP address of the interface that was used for the authentication.

For example:

```
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
Passwords must not contain spaces or equal symbols ('=').

In high availability configurations (cluster)

When the BGP dynamic routing protocol is used in an SNS firewall cluster, and to allow the BGP neighbor to shut down a BGP session properly during a switch in the cluster, it would be helpful to define a BFD instance in the BIRD configuration:

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

In this example:

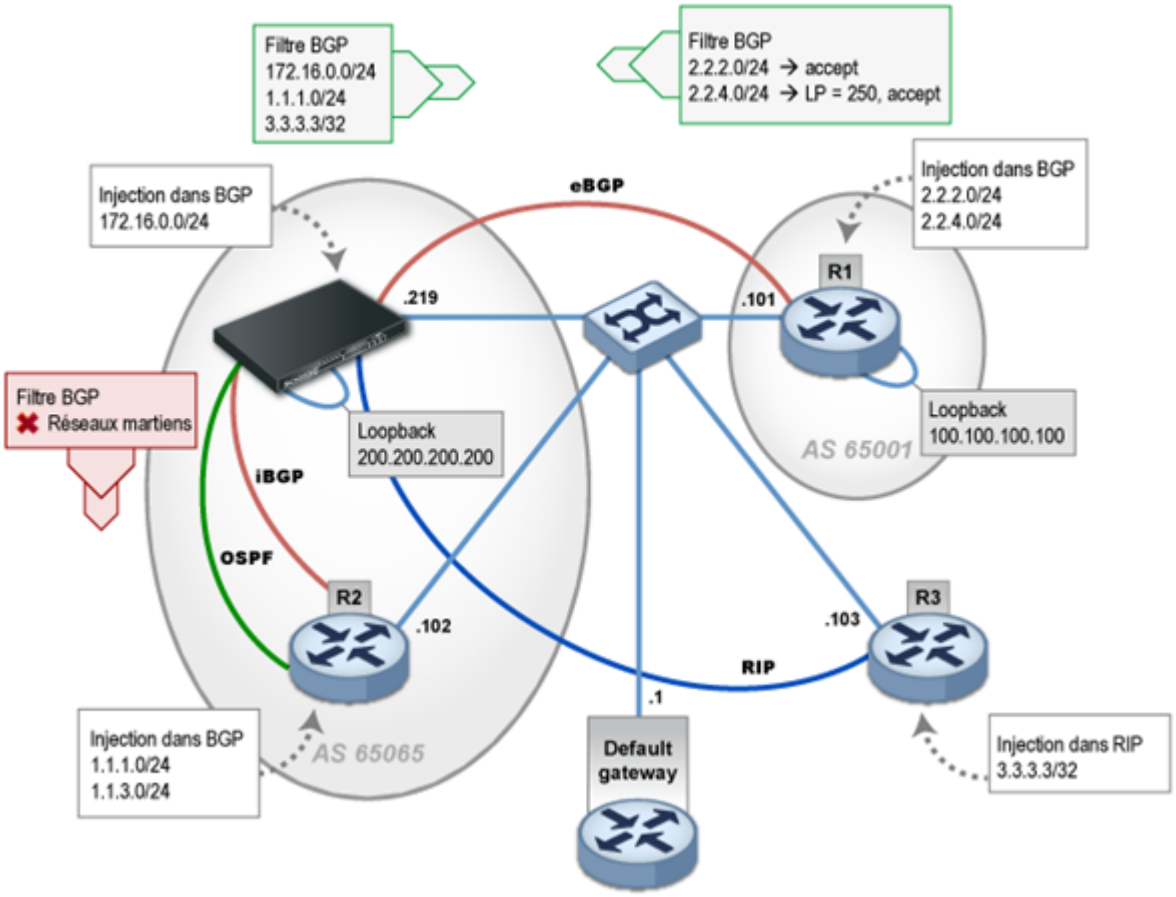
```
protocol bfd mybfdsession {
        neighbor 100.100.100.100;
}
```

And to name this instance in the BGP configuration accordingly:

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

Advanced configuration

Advanced configuration is implemented here. In addition to this configuration connecting the three individual configurations, it also contains an iBGP link that was set up simultaneously with the OSPF link.



The client's network includes routers R2, R3 and the Stormshield Network firewall. Router R1 is an external BGP neighbor. This network represents a realistic architecture setup, except for the fact that all routers are physically connected through a single LAN.

A standard filter policy is implemented to:

- Announce only outbound public BGP networks,
- Avoid advertising internal networks or martians in the internal BGP,
- Tag one of the routes that were learned in eBGP, with a local-preference value of 250. This measure is generally implemented to monitor load balancing among several eBGP neighbors,
- Announce only one default route in OSPF,
- Announce only one default route in RIP,

The networks that are announced by routers R2 and R3 are respectively announced over BGP and RIP. The use of OSPF to announce the default route is for illustration purposes only.

BIRD configuration

The following is the equivalent configuration file in 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
You are advised to set the value of the parameter `priority 0` in the *interface* section of the OSPF node configuration, in order to disable the firewall's participation in elections for Designated Router/Backup Designated Router roles.

Allowing RIP, BGP and OSPF protocols in the filter policy

By referring to the sample filter policies presented in the individual RIP, BGP and OSPF configurations, add filter rules to allow RIP, BGP and OSPF routing traffic to and from the firewall.

Checking the proper operation of dynamic routing

Stormshield Network firewall routing table

```
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)
```

To check the value of the local-preference on route 2.2.4.0/24, the details of the routes belonging to the instance of the router1 protocol are shown:

```
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

You will notice that the default route is also announced here:

```
@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:~$
```

When this traffic has to be symmetrically routed, for example in the case of NAT, the BIRD configuration must be adapted so that the firewall can be announced as the next-hop. The configuration can be changed in the "default_ok" filter, which is used to announce the default route to R3 over RIP, and to R2 over 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;
  }
}
```

To impose a gateway other than the firewall itself, this directive must be used:

```
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:~$
```

In high availability configurations (cluster)

When the BGP dynamic routing protocol is used in an SNS firewall cluster, and to allow the BGP neighbor to shut down a BGP session properly during a switch in the cluster, it would be helpful to define a BFD instance in the BIRD configuration:

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

In this example:

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

And to name this instance in the BGP configuration accordingly:

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

Migrating a dynamic routing configuration from BIRD v1 to BIRD v2

As of SNS version 4.8.1, the BIRD v2 dynamic routing engine will be supported, and replaces BIRD v1, which has become obsolete.

When you upgrade to SNS version 4.8.1 a firewall with a configuration that initially used BIRD v1 dynamic routing, BIRD v1 will remain active even after the firmware has been updated.

This is because your configuration cannot be automatically transferred from BIRD v1 to BIRD v2, as the syntax used in the BIRD v2 dynamic routing configuration file is different from the syntax in BIRD v1.

One of the major changes is that in BIRD v2, IPv4 and IPv6 dynamic routing settings have been grouped into a single *bird.conf* file, unlike BIRD v1, which uses two separate files: the same *bird.conf* file for IPv4 and the *bird6.conf* file for IPv6.

The **Dynamic routing** module in SNS versions 4.8.1 and higher has been designed to assist you in the migration operation.

IMPORTANT
If your SNS firewall pool is managed by an SMC server, it will no longer be possible to manage dynamic routing on your firewalls in 4.8.1 versions and higher from SMC in version 3.6 and below.

Understanding the Dynamic routing module

Go to **Configuration > Network > Dynamic routing.**

This module contains three tabs in which either version of BIRD can be enabled/disabled and configured.

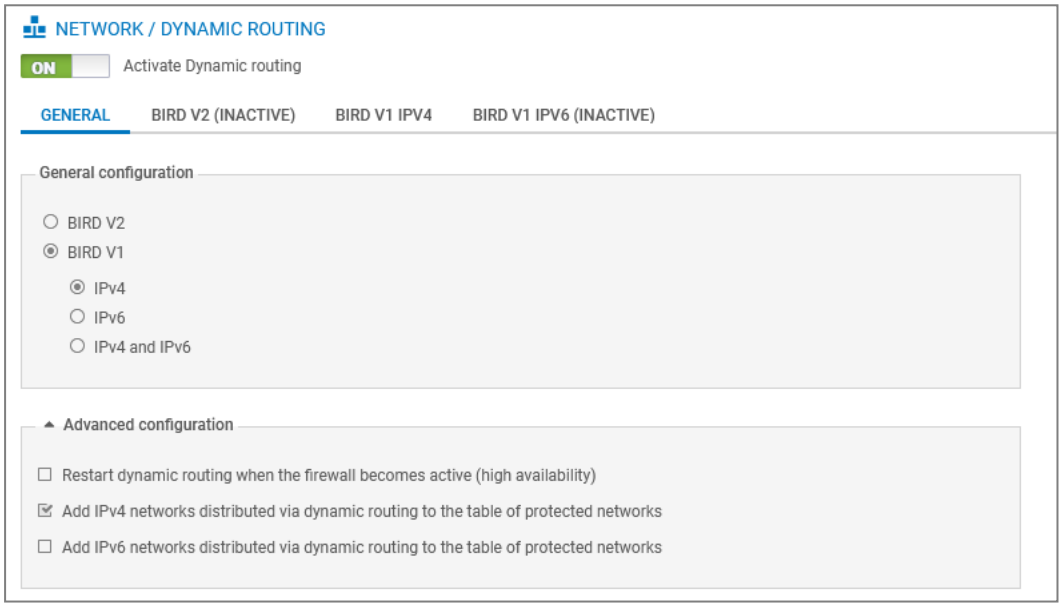
NOTE
When one version of BIRD is disabled, the corresponding configuration tab will show the suffix " (INACTIVE)".
Example: BIRD v2 (INACTIVE).

General tab

This option allows you to enable/disable the desired version of the BIRD dynamic routing engine.

After SNS firewalls in a version lower than SNS 4.8.1 are updated to SNS version 4.8.1 or higher, the configuration will be as follows:

- **BIRD v2**: this radio button is selected by default.
- **BIRD v1**: this radio button will be selected if the firewall was initially configured only in IPv4, and if its IPv4 BIRD v1 configuration was active prior to the firmware update.
  The following radio buttons will appear only if the firewall was initially configured in IPv4 and IPv6:
  - **IPv4**: this radio button is selected for firewalls on which only an IPv4 BIRD v1 configuration was active prior to the firmware update.
  - **IPv6**: this radio button is selected for firewalls on which only an IPv6 BIRD v1 configuration was active prior to the firmware update.
  - **IPv4 and IPv6**: this radio button is selected for firewalls on which IPv4 and IPv6 BIRD v1 configurations were active prior to the firmware update.



IMPORTANT
If you want the routes that were learned by BIRD to be automatically added to the table of protected networks, thereby preventing these networks from wrongly raising antispoofing alerts, select these checkboxes (depending on your configuration):

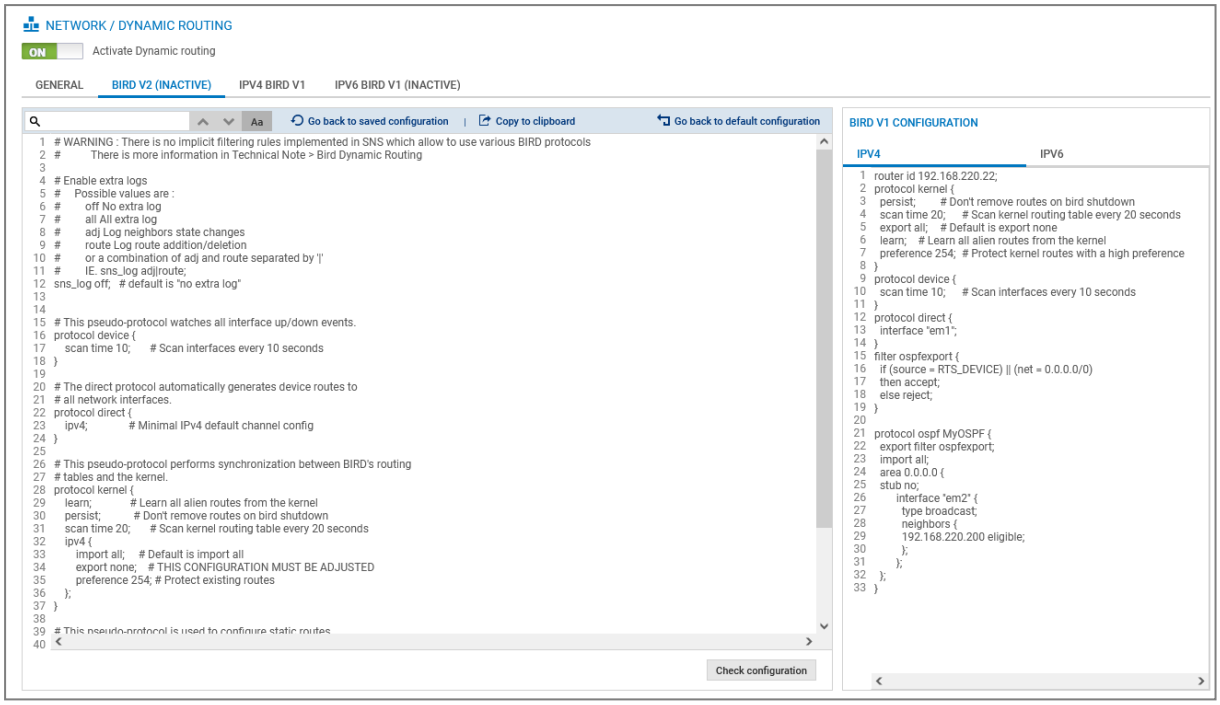
- **Add IPv4 networks distributed via dynamic routing to the table of protected networks**.
- **Add IPv6 networks distributed via dynamic routing to the table of protected networks**.

BIRD v2 tab

This tab shows:

- On the left side of the screen: a minimalist BIRD v2 configuration frame containing the basic mandatory sections,
- On the right side of the screen: the firewall's original BIRD v1 configuration (IPv4 and/or IPv6).

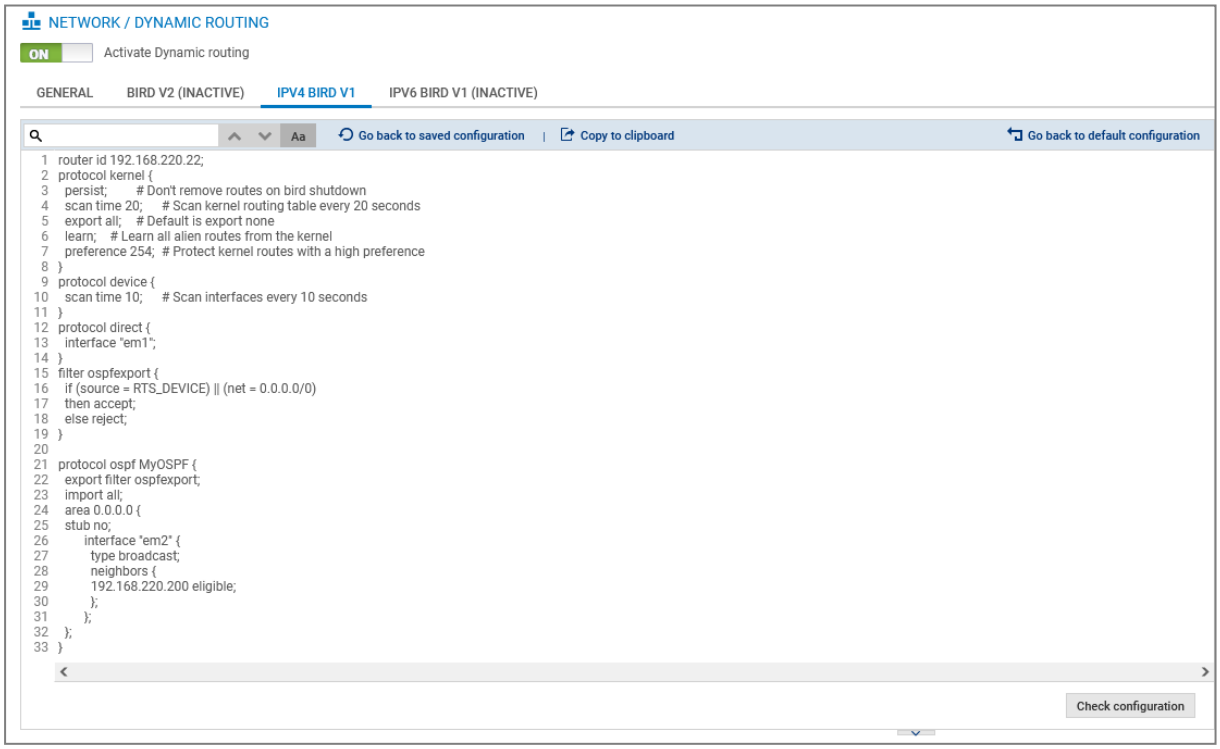
This section also allows you to modify the firewall's BIRD v2 configuration and validate it.



IPv4 BIRD v1 tab

This tab shows the original configuration on the firewall for the IPv4 dynamic routing managed by BIRD v1.

This section also allows you to edit and validate the configuration.



Optional IPv6 BIRD v1 tab

This tab shows the original configuration on the firewall for the IPv6 dynamic routing managed by BIRD v1.

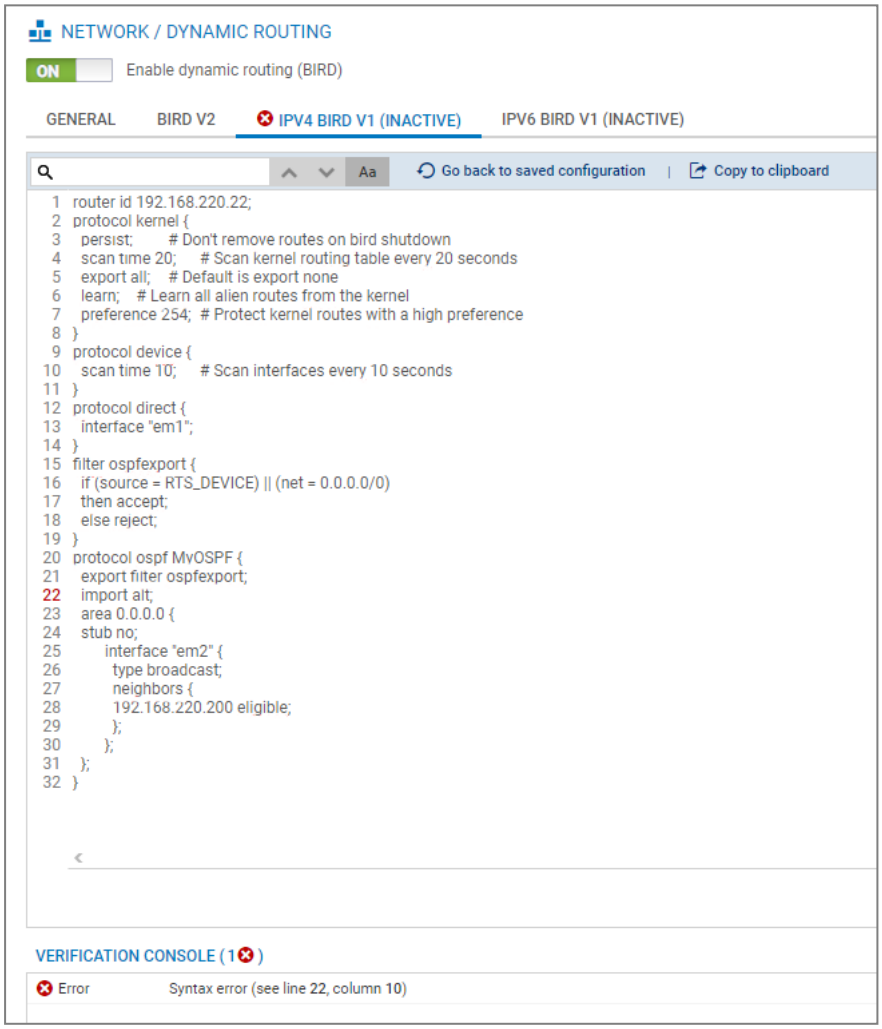
This section also allows you to edit and validate the configuration.

It looks exactly like the **IPv4 BIRD v1/IPv4 BIRD v1 (INACTIVE)** tab.

Verification console

When you click on the **Check configuration** button in one of the BIRD configuration tabs shown below, the verification console located at the bottom of the screen shows the syntax errors encountered, if any.

Errors are identified in the console by their line numbers and column numbers. Line numbers that contain errors are also highlighted in red in the configuration:



Migrating from BIRD v1 to BIRD v2

Stormshield recommends that you follow the method below:

Preparing the BIRD v2 configuration

1. Go to the **BIRD v2 (INACTIVE)** tab.
2. By following the BIRD v2 configuration syntax, transpose the information from your BIRD v1 configuration (window on the right) to the BIRD v2 configuration (window on the left) in stages. As a reminder, in BIRD v2, IPv4 and IPv6 dynamic routing settings have been grouped into a single *bird.conf* file, unlike BIRD v1, which uses two separate files: the same *bird.conf* file for IPv4 and the *bird6.conf* file for IPv6.

   > **NOTE**
   > If you require assistance in this task, refer to the available resources, notably the BIRD v2 user guide published by BIRD, and the BIRD 1.6 to BIRD 2.0 transition notes.

3. While you are making changes to the configuration, click regularly on the **Check configuration** button after the changes are made.
   At the bottom of the screen, the consistency checker will show you the syntax errors found in the BIRD v2 configuration.
   You cannot save configurations that contain syntax errors.
4. When changes to the BIRD v2 configuration are made and validated (no errors shown in the **Verification console**), save the configuration by clicking on **Apply**, then **Save**.
   This operation creates a version of the BIRD v2 configuration that can be restored. If changes are made later to the BIRD v2 configuration, and you are unable to fix a configuration/syntax issue, the configuration at this stage can be restored by clicking on **Go back to saved configuration**.
5. When the migration operation is complete, and you have saved your BIRD v2 configuration, you can enable BIRD v2 to check whether dynamic routing is running properly.

Checking the operation of dynamic routing

After you have enabled BIRD v2, if you detect issues with the way dynamic routing is running:

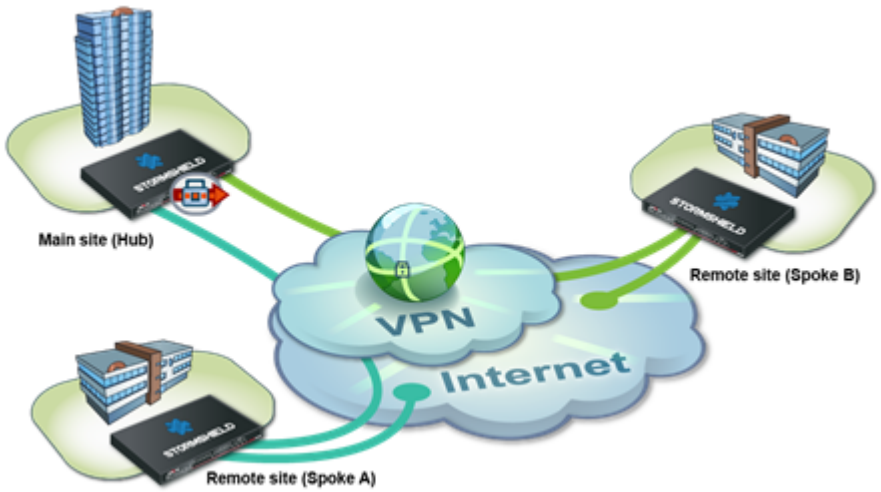
1. In the **General** tab, disable BIRD v2 and enable BIRD v1 again to return to the state of the configuration before BIRD was migrated.
   You can then fix your BIRD v2 configuration, while BIRD v1 dynamic routing configuration remains active.
2. Enable BIRD v2 again once the configuration has been fixed.

These operations can be repeated as often as required.

Appendix A: Hub and Spoke VPN tunnels routed via BGP

The following is an example of BGP dynamic routing in a hub and spoke, or star, VPN topology.

Tunnel configuration



For details on how to configure the hub and spoke IPsec policy, refer to **Case no. 1: internal traffic via IPsec tunnels** in the IPsec VPN - Hub and Spoke Configuration technical note.

In our example, the way the settings differ from the procedure is in the configuration of traffic endpoints through virtual interfaces, instead of remote networks in the IPsec policy:

Main site

TunnelA

Local network:	ipsec1 interface (172.16.0.1)
Peer:	Site_SpokeA
Remote network:	Remote_tunnelA (172.16.0.2)

TunnelB

Local network:	ipsec2 interface (172.16.0.5)
Peer:	Site_SpokeB
Remote network:	Remote_tunnelB (172.16.0.6)

Spoke A

Local network:	ipsec1 interface (172.16.0.2)
Peer:	Site_FW_Hub
Remote network:	Remote_tunnelA (172.16.0.1)

Spoke B

Local network:	ipsec1 interface (172.16.0.6)
Peer:	Site_FW_Hub
Remote network:	Remote_tunnelB (172.16.0.5)

BGP configuration of the main site (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;
}
```

BGP configuration of satellite site 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;
}
```

BGP configuration of satellite site 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;
}
```

Verification of routing tables

Routing table on the main site (Hub):

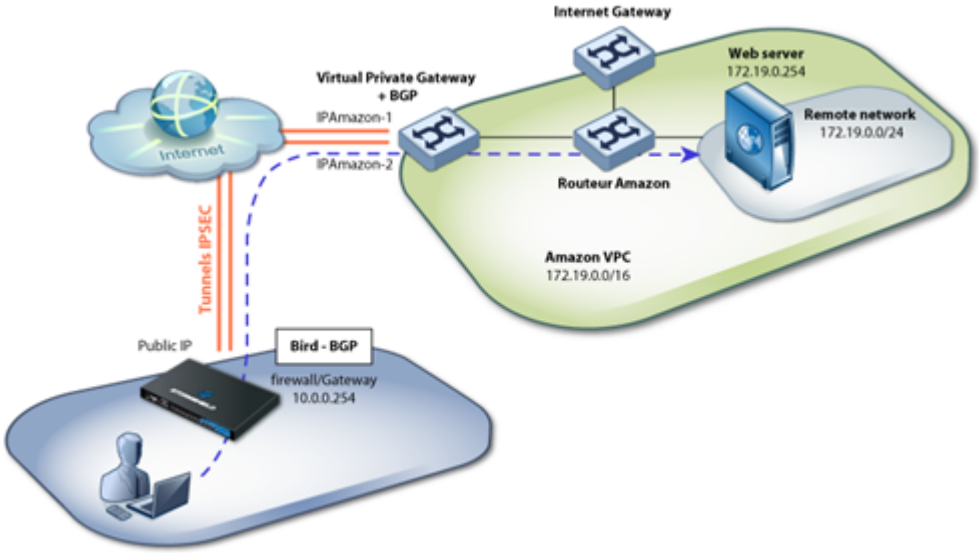
```
bird> show route
0.0.0.0/0              via 10.60.0.254 on em0 [kernel1 10:16] * (254)
10.60.3.127/32         dev lo0 [kernel1 10:16] * (254)
192.168.0.0/24         dev em1 [direct1 10:16] * (240)
192.168.1.0/24         dev em2 [direct1 10:16] * (240)
192.168.1.0/24         via 172.16.0.2 on enc1 [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)
```

Routing table on 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)
```

Appendix B: Connectivity with Amazon VPC

The aim here is to connect a local network to an Amazon VPC (virtual private cloud). To do so, Amazon offers the possibility of creating two routed tunnels between the local firewall and the Amazon cloud, and then routing this traffic over BGP.



Amazon VPC configuration

Follow the steps below:

1. Create an Amazon VPC.
2. Create a subnet in this VPC.
3. Configure routing in this VPC.
4. Create a dynamic VPN connection to this firewall through the Amazon Virtual Private Gateway object.
5. Create ACLs to allow local traffic to the web server.
6. Routing: allow routes to be announced to the VPC routing table.

Excerpt of the configuration help provided by Amazon during the configuration of the service:

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

Outside IP Addresses:
     • Customer Gateway: Firewall/Gateway public IP,
     • 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
```

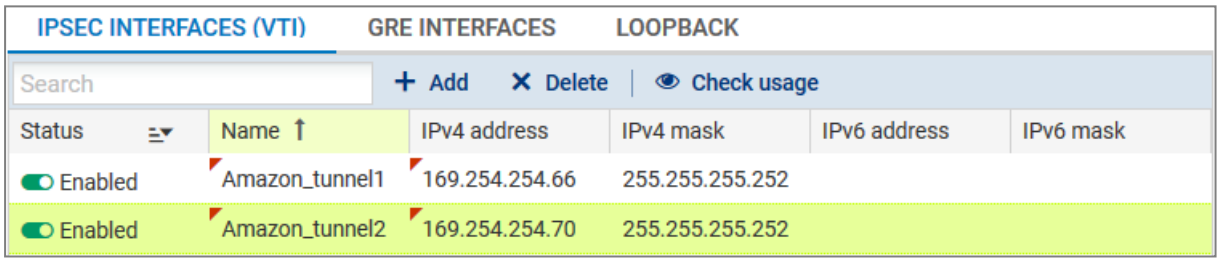
```
inside IP addresses, to exchange routes from the VPC to your home network.
Each BGP router has an Autonomous System Number (ASN). Your ASN was
provided to AWS when the Customer Gateway was created.

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.
```

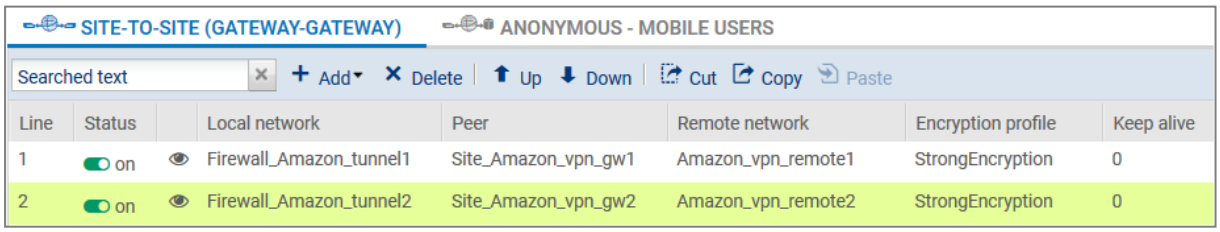
Tunnel configuration

In **Configuration** > **Network** > **Virtual interfaces**, the *IPsec interfaces* tab allows you to define the interfaces in question:



In **Configuration** > **VPN** > **IPsec VPN**, *Site-to-site (gateway-gateway)* tab, you can define the tunnels below by using the following objects:

- **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.



BGP configuration

Here, we are choosing to export only the network 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;
        };
}
```

Further reading

Additional information and answers to questions you may have about BIRD dynamic routing are available in the Stormshield knowledge base (authentication required).

documentation@stormshield.eu