



# Deployment Guide for MSSPs

Secure SD-WAN



DEFINE / DESIGN / **DEPLOY** / DEMO



# Table of Contents

<b>Change Log</b>	4
<b>Introduction</b>	5
<b>Deployment overview</b>	6
Model devices	6
Device groups and provisioning templates	7
Jinja templates	8
Automation	8
Multitenancy	9
<b>Design overview</b>	11
Example project	11
Routing design methods	13
BGP per overlay	13
BGP on loopback	14
BGP loopback design advantages	15
BGP on loopback design limitations	16
Recommendations for choosing a routing method	16
<b>Preparing Jinja templates</b>	17
Understanding template structure	17
Defining the Project template	18
Example project template	20
<b>Building a project foundation in FortiManager</b>	24
Device groups	24
System templates	26
Certificate templates	27
Jinja CLI templates	28
Importing Jinja CLI templates	29
Creating CLI template groups	31
Assigning CLI template groups	32
SD-WAN templates	32
Creating Edge SD-WAN templates	33

Creating Hub SD-WAN templates .....	39
Static route templates .....	41
Creating a static route template .....	41
Adding a static default route .....	42
Assigning the static route template .....	42
Adding a static route for the next-hop gateway .....	43
Firewall policies .....	45
Creating common elements .....	45
Creating the Edge policy package .....	46
Creating the Hub policy package .....	48
<b>Deploying sites</b> .....	51
Creating a model device .....	52
Setting meta field values .....	53
Generating a certificate .....	54
Installing the configuration .....	55
Installing policy packages .....	56
Onboarding devices .....	57
<b>Additional topics</b> .....	59
SD-WAN routing logic .....	59
Best route mode for SD-WAN .....	60
SD-WAN as default route .....	61
Excluding Traffic from SD-WAN .....	63
<b>Appendix A - Products used</b> .....	65
<b>Appendix B - External resources</b> .....	66

# Change Log

Date	Change Description
2022-08-19	Initial release.

# Introduction

This document will guide you through a deployment of the Fortinet Secure SD-WAN Solution that consists of fully functional FortiGate (FGT) devices deployed on every site and centrally managed by FortiManager (FMG) and FortiAnalyzer (FAZ).

Our aim is to present an approach with the following important characteristics:

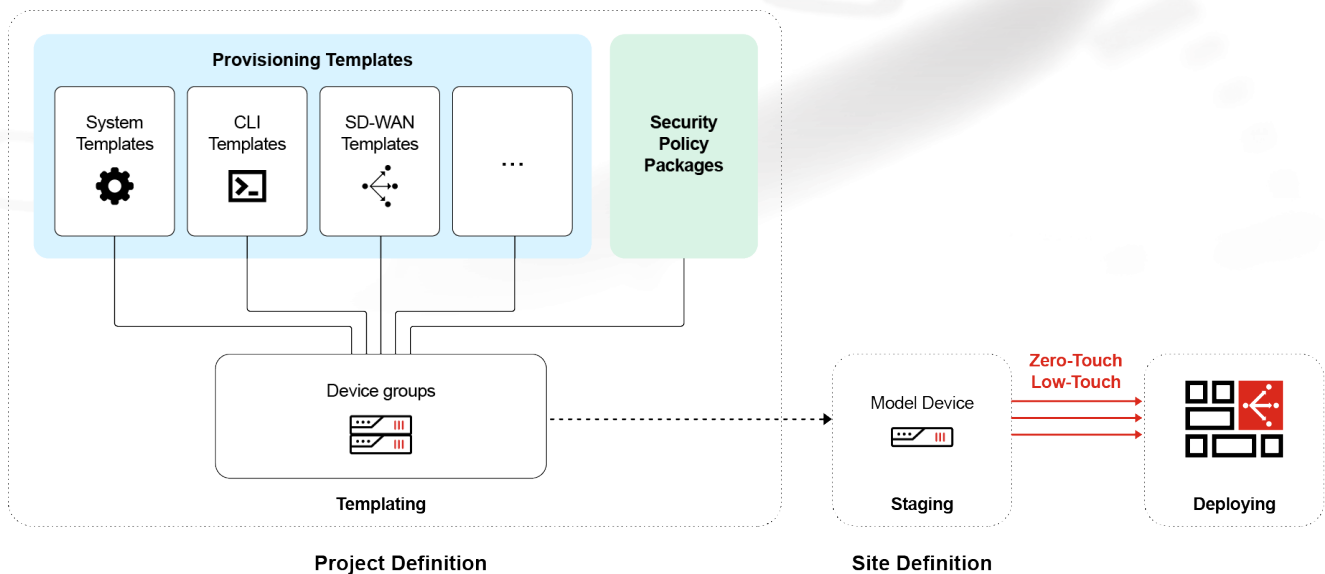
- **Generic:** suitable for a wide variety of topologies with a mix of different site types
- **Flexible:** allows (but not requires!) every detail to be customized
- **Reusable:** provides a high degree of similarity (and re-usability) between different projects
- **Automation-friendly:** suitable for fully automated provisioning using REST API
- **IaC-friendly:** includes plain-text project descriptions in a declarative (also considered implementation-agnostic) language, which is both human and machine-readable, and allows for easy review, traceability, version control, and more

This approach is recommended for Managed Service Providers, who are the main target audience for this document. However, this approach may benefit any type of customer looking for the characteristics mentioned above.

This document contains the following topics:

- [Deployment overview on page 6](#)
- [Design overview on page 11](#)
- [Preparing Jinja templates on page 17](#)
- [Building a project foundation in FortiManager on page 24](#)
- [Deploying sites on page 51](#)
- [Additional topics on page 59](#)

# Deployment overview



This section describes the following important concepts related to the workflow:

- [Model devices on page 6](#)
- [Device groups and provisioning templates on page 7](#)
- [Jinja templates on page 8](#)
- [Automation on page 8](#)
- [Multitenancy on page 9](#)

## Model devices

You can use multiple methods to deploy FortiGate devices that are centrally managed by FortiManager.

One option could be the following:

1. Manually deploy a new FGT with basic underlay configuration that is necessary to communicate with FMG.
2. In FMG, onboard the FGT by using the device discovery method.
3. In FMG, make the necessary configuration, and push the configuration to the new FGT.

While this method can be satisfactory in some environments, we recommend following a different approach, which relies on the concept of a **model device** in FMG. The model device behaves similar to any managed FGT device, except that there is no *real* FGT device behind it. Therefore, the following steps are required:

1. In FMG, create a model device for the new FGT that needs to be deployed.
2. In FMG, apply all the necessary configuration to the model device.
3. Link the *real* FGT device to the model device.

The first two steps are done locally on FMG without any communication with the real FGT device. Hence, they can be done during preparation stage, when the real FGT device might not even exist yet.

The last step is where the configuration is pushed to the real FGT device, making it part of the SD-WAN deployment. The last step can be done several ways:

- **Zero-Touch Provisioning (ZTP) using FortiDeploy** allows linking the real FGT device to the model device with minimal interaction with the FGT device itself:
  - Once the new (and not yet configured) FGT device is plugged in, it "calls home" to obtain the location of FMG.
  - FGT contacts FMG, which authorizes the FGT and maps the FGT to the corresponding model device by using the serial number.
  - FMG pushes the model device configuration to the FGT device.
  - Once this operation is complete, the FGT is fully deployed and managed by FMG.
- **Zero-Touch Provisioning (ZTP) using DHCP Option** is identical to the previous method, except for how the FGT obtains the location of FMG. Instead of "calling home", the new (and not yet configured) FGT device receives the information from the local DHCP server by using a special DHCP Option 240. But from the FMG perspective, the sequence of events remains the same.
- **Low-Touch Provisioning** requires a customer to manually configure FMG details on the new FGT device:
  - Once the new (and not yet configured) FGT device is plugged in, the user connects to it and manually enters the details of FMG.
  - FGT then contacts FMG, and the rest of the process remains identical to the previous methods. Here again, from the FMG perspective, the sequence of events remains the same.



Detailed instructions for the above methods are outside the scope of this document.

The model device approach offers multiple advantages compared to the traditional device discovery method:

- The workflow does not depend on whether ZTP is used (or on what ZTP method is used). Different sites can be onboarded using different methods, but from the FMG perspective, the process remains the same for all of them.
- Model devices are local objects inside FMG. They can be safely deleted, in case any configuration mistakes have been made. Since this configuration is not pushed to any "real" FGT device until the very last step, there is no need for rollback.
- For the same reason, working with model devices is *safer*. Interim configuration will not cause service disruption, since only the final configuration is pushed to the "real" FGT device at the very last step.

For the above reasons, we will be focusing on the model device approach throughout this document.

## Device groups and provisioning templates

As described in [Model devices on page 6](#), a model device behaves similar to any managed device in FMG. Therefore, the configuration could be done on a per-device basis, using Device Manager.

However, we **strongly discourage** you from using a per-device method when configuring your Secure SD-WAN Solution. Interactive per-device configuration using Device Manager quickly becomes unacceptable, as the number of sites grows. It cannot be easily replicated to other devices (or to other environments).

For this reason, we recommend relying on the two crucial entities in the FMG:

- FGT devices should be grouped into **device groups**, representing the different types of sites in your project.

- The right set of **provisioning templates** should be assigned to each device group. These include System Templates, Certificate Templates, SD-WAN Templates, CLI Templates, and so on, and will be described in more detail throughout this document. The templates generalize the configuration that will be provisioned on your FGT devices.

The amount of ad-hoc per-device configuration should be reduced to the minimum: it should be limited mainly to setting per-device meta fields (variables) and assigning the device to the right device group, which will automatically lead to the assignment of the right set of provisioning templates.



The use of device groups and provisioning templates is always encouraged, whether or not you are planning to use zero-touch provisioning. Good reusable templates are the key for a successful large-scale Secure SD-WAN deployment!

## Jinja templates

To configure **Underlay**, **Overlay**, and **Routing** pillars, we will mostly use the new Jinja Template engine integrated into FMG 7.0.x.

Although this is just one of the available kinds of the provisioning templates, it deserves a separate section.

With Jinja templates, you can describe your project in a human-readable declarative (also considered implementation-agnostic) language and create what we will refer to as **a Project template** written in Jinja. Next, this Project template will be used by a set of Jinja CLI Templates to generate the necessary FortiOS (FOS) configuration.

Although these Jinja CLI Templates can be written from scratch, we provide a ready-to-use set of templates, producing our best practice configuration for SD-WAN deployments. In this document we will demonstrate its use.



The latest version of the templates can be found in our dedicated GitHub repository. See [Appendix B - External resources on page 66](#).



Any plain text editor can be used to view and edit Jinja templates. Many editors conveniently provide syntax highlighting. Here are some popular editors to consider:

- [Atom](https://atom.io/) (<https://atom.io/>) is available on macOS, Windows, and Linux.
- [Visual Studio Code](https://code.visualstudio.com/) (<https://code.visualstudio.com/>) is available on macOS, Windows, and Linux.
- [Notepad++](https://notepad-plus-plus.org/) (<https://notepad-plus-plus.org/>) is available on Windows.

## Automation

FortiManager (FMG) provides a comprehensive REST API that allows complete automation of the deployment. Every action described in this document can be automated with an API call made using an automation framework of your choice, such as Ansible, Python, and so on.

As a result, it is possible to deploy our entire Secure SD-WAN Solution in a fully automated, unattended manner. Interactive login to FMG is not required.



While a detailed walkthrough is outside the scope of this document, it is useful to highlight some of the benefits of the deployment done this way:

- Infrastructure as Code (IaC). All templates and other FMG objects can be imported from an external repository, using automation. This allows you to replicate the environment quickly from scratch, for example, to maintain consistency between staging and production environments.
- Model device creation can be automated as well, including filling in the right meta field values per site, assigning the right set of templates, and performing any other required configuration. For example, model device creation can be triggered by an external Onboarding Portal, to which the remote site operator connects, in order to fill in the necessary details of the new FGT device (such as its serial number).
- For Managed Secure Service Providers (MSSPs), custom Onboarding Portals provide a valuable opportunity *to speak their end-customer's language*: the UI of those portals can be designed with consideration to a particular end-customer's business, using the relevant language, rather than generic SD-WAN terminology.
- Needless to say: automation saves time, minimizes human mistakes and becomes indispensable in large-scale deployments.



As a starting point, we provide an automation collection tailored to this guide that can be used with Postman software. See [Appendix B - External resources on page 66](#).

---

## Multitenancy

Each component of our solution offers flexible multitenancy options. FortiGate devices (FGT) can be shared between multiple tenants using *Virtual Domains (VDOMs)*. FortiManager (FMG) and FortiAnalyzer (FAZ) use *Administrative Domains (ADOMs)*.

When our solution is offered as a Managed Service, the following deployment options are typically considered by the Service Providers (MSSPs):

- Edge FGTs are deployed on end-customer premises, and they are dedicated to those end-customers (that is, no multitenancy).
- Hub FGTs:
  - Option 1: Deployed on MSSP premises, multitenant, VDOM per end-customer
  - Option 2: Deployed on MSSP premises, dedicated to end-customer, no multitenancy (usually virtual)
  - Option 3: Deployed on end-customer premises (Enterprise design), no multitenancy
- FMG/FAZ:
  - Option 1: Deployed on MSSP premises, multitenant, ADOM per end-customer
  - Option 2: Deployed on MSSP premises or in public cloud, dedicated to end-customer, no multitenancy (virtual)

When MSSP offers a fully managed service, end-customers might not even require access to the devices. But quite often certain level of access is provided. The following options are worth considering:

- If end-customers must have a certain level of access to FGT devices (often read-only), a dedicated customer VDOM can be created. The management ("root") VDOM will be accessible by the MSSP only.
- Central management and monitoring:
- End-customer may be granted direct FMG/FAZ access, to their respective ADOM only (or to the entire instance, when it is dedicated to end-customer).

## MULTITENANCY

---

- Alternative option is to add FortiPortal to the solution, which is a comprehensive self-service portal designed for the end-customers.
- Yet another option is to use a custom MSSP Portal developed in-house. This portal can communicate to FMG/FAZ using REST API, thanks to the comprehensive automation support described above.

The choice of multitenancy model has minimal impact on the rest of this document. It is only important to make sure that the configuration is done in the right ADOM.



# Design overview

The basic building block of our SD-WAN solution is a Hub-and-Spoke topology, which we refer to as a **region**. The solution can include one or more regions, each of them typically served by one or two Hubs.

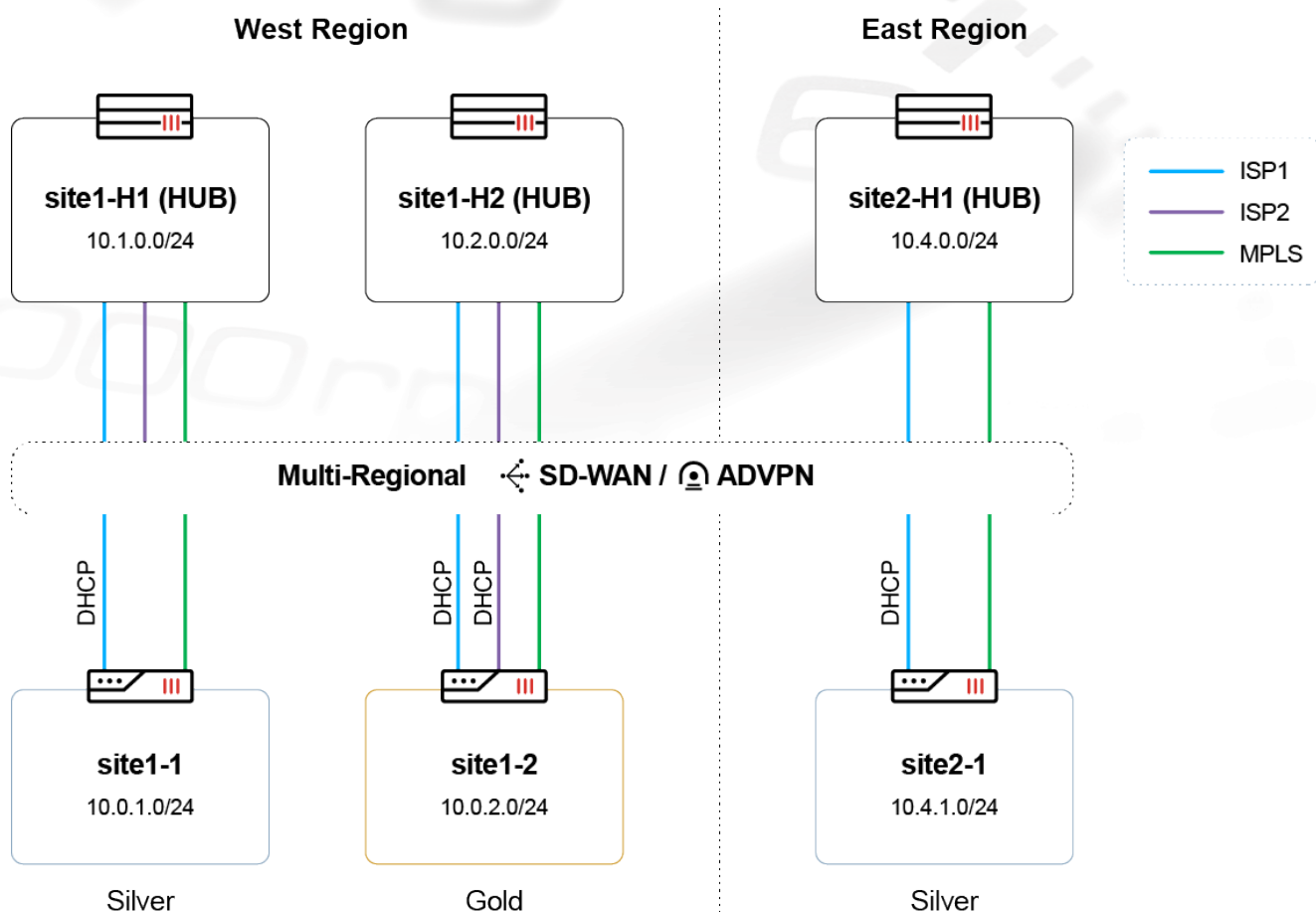
As mentioned in the [Introduction on page 5](#), we are going to describe a generic deployment method suitable for SD-WAN solutions with an arbitrary number of regions, including a mixture of Single-Hub and Dual-Hub regions, sites of different sizes connected to different number of WAN transports, served by different Edge device models, and so on.

The following sections help describe the design of the managed Fortinet Secure SD-WAN solution:

- [Example project on page 11](#)
- [Routing design methods on page 13](#)

## Example project

For a better readability, we will be using the following SD-WAN project as an example throughout this document:



As can be seen on the diagram, the project is comprised of two regions:

- **West Region** is served by two Hubs.
- **East Region** is served by a single Hub.

The Edge sites offer two levels of connectivity, which are referred to as **profiles**:

- **Silver** profile means two WAN links: one Internet connection (ISP1) and one MPLS.
- **Gold** profile means three WAN links: two Internet connections (ISP1, ISP2) and one MPLS.

To summarize, each of the Edge sites in our example will have slightly different properties:

Site	Region	Profile	Serving Hubs	WAN links
site1-1	West	Silver	2	2
site1-2	West	Gold	2	3
site2-1	East	Silver	1	2

The Hubs serving the two regions are as follows:

Site	Region	Profile	WAN links
site1-H1	West	Gold	3
site1-H2	West	Gold	3
site2-H1	East	Silver	2

It can also be seen that both Internet connections in our project have DHCP servers, while the MPLS connection *does not*. As such we must manually provision the interface IP address and the next-hop gateway for the MPLS WAN link on each site.

In addition, as it typically happens, each Edge device will act as a DHCP server for its local LAN.

In the following sections we will demonstrate how you can use our generic deployment method to deploy such a mixture of different sites and transports.

## Routing design methods

Currently we provide two main routing design methods:

- [BGP per overlay on page 13](#)
- [BGP on loopback on page 14](#)

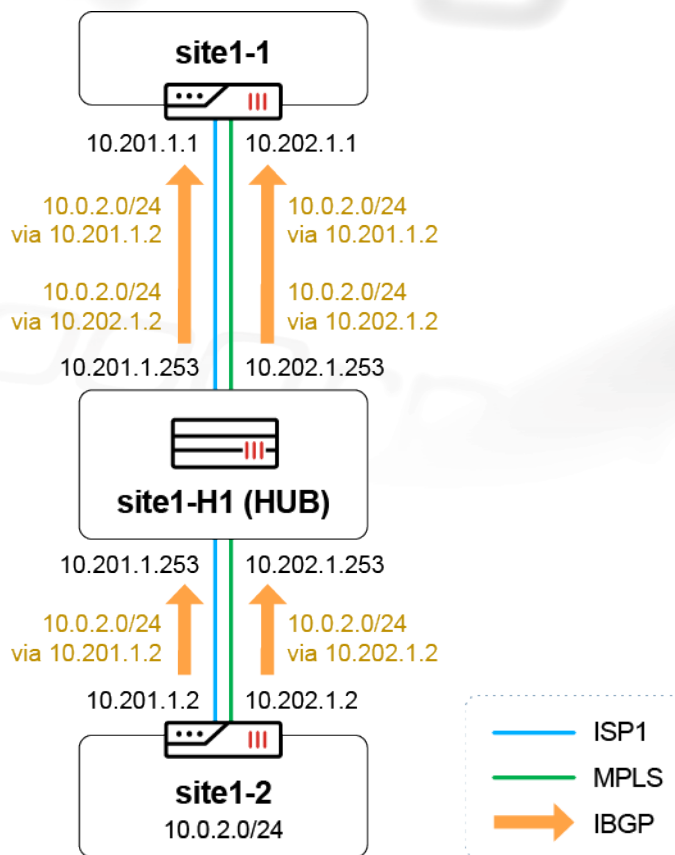
In both routing design methods, the Hubs act as BGP Route Reflectors, propagating the routes across all the Edge devices in the network.

This section also covers the following topics:

- [BGP loopback design advantages on page 15](#)
- [BGP on loopback design limitations on page 16](#)
- [Recommendations for choosing a routing method on page 16](#)

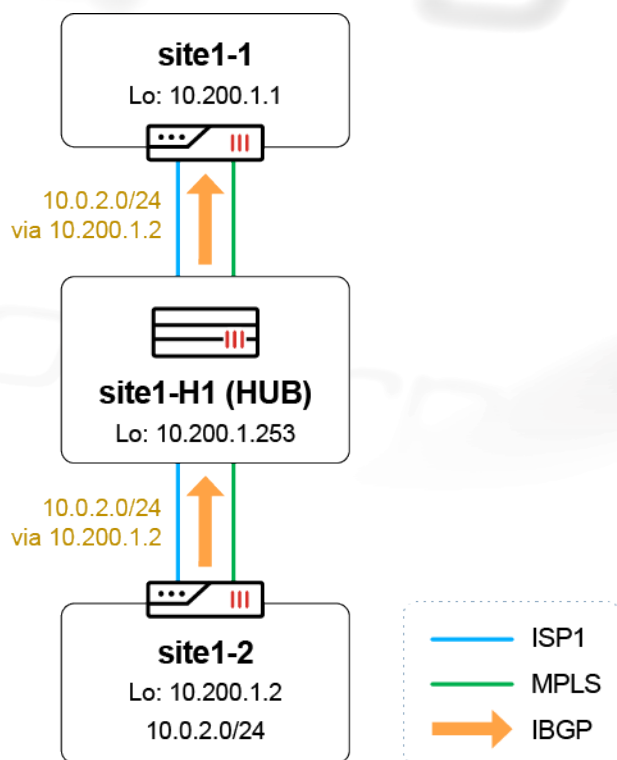
### BGP per overlay

The BGP per overlay method is the traditional routing design for our SD-WAN/ADVPN deployments. With BGP per overlay, a separate IBGP session is established over each overlay between an Edge device and a Hub. This IBGP session is terminated on the tunnel IP of both sides. For each LAN prefix, multiple BGP routes are generated (one route per overlay), and all these routes are propagated across the network.



## BGP on loopback

The BGP on loopback method is a new alternative supported for our SD-WAN/ADVPN deployments, starting from FOS 7.0.4. With this routing design, a single IBGP session is established between an Edge device and a Hub. This IBGP session is terminated on the loopback interface on both sides, but the routes are recursively resolved through all available overlays. For each LAN prefix, a single BGP route is generated and propagated across the network.



## BGP loopback design advantages

Compared to the traditional BGP per overlay design, the new BGP on loopback design offers the following advantages:

1. **Significantly improved scalability:** This is particularly true on the Hubs, since they must reflect a much smaller number of routes between a much smaller number of IBGP peers.



Note that the problem of BGP Route Reflection is a "square problem" ( $O(n^2)$ ): the number of routes to reflect equals the number of routes multiplied by the number of peers (except for the peer that originally advertised the route). Therefore, reducing both these numbers significantly improves the scalability of the route reflector.

Additionally, since only a single BGP route is generated for each LAN prefix, BGP ADD-PATH functionality is no longer required. Thus, the Hubs do not inflate the number of routes, neither when reflecting them between the Edge devices nor when advertising them to other regions.

2. **Significantly simplified configuration:** The difference can be seen even in a single region with a simple topology, but it becomes more apparent in topologies with multiple Internet links, and it is nearly unmissable when the solution expands to multiple regions.



The difference comes not only because of fewer BGP neighbors to configure, but also because the configuration becomes much simpler: no BGP ADD-PATH, no need to plan and configure tunnel IP addresses, no potential need for duplicate route filtering, and so on.

## BGP on loopback design limitations

The following limitations currently exist in the new BGP on loopback design:

- 1. Reduced support for Hub-to-Edge SD-WAN:** When sessions are originated behind the Hub, it is generally not possible to guarantee that they select the preferred or healthy overlay on their way to the Edge devices. They will follow the conventional routing, potentially selecting an unhealthy or a backup overlay.
- 2. Suboptimal switchover with segregated transports:** In topologies that include several transports that are physically segregated from each other (the most common example being Internet and MPLS), in certain failure scenarios, the traffic between two Edge devices might not trigger a direct ADVPN shortcut during switchover, for example, consider a classical dual-overlay topology (Internet + MPLS, with Internet being the preferred transport). When an Internet link is down on the remote Edge device, the traffic might not build a backup end-to-end shortcut via the MPLS, but rather it would flow via the Internet overlay between the originating Edge and the Hub, at which point the Hub would switch it to the MPLS overlay.



Despite the above limitations, the traffic will nevertheless reach its destination. Additionally, these limitations might not apply in certain simple scenarios. We recommend contacting your Fortinet representatives for more details in the context of your project.

## Recommendations for choosing a routing method

The following table summarizes our current recommendation on choosing the right routing design method for your project. Please keep in mind that this table should be used only as a general guidance tool. We strongly advise that you contact your Fortinet representatives to obtain a more specific recommendation for your project!

Requirement	Recommended Design
Sites with multiple Internet links and/or different number of Internet links per site	BGP on loopback
Large-scale deployments and/or Multi-regional deployments	BGP on loopback
Inter-regional ADVPN, Cross-overlay (for example, cross-ISP) ADVPN	BGP on loopback
Hub-to-Edge SD-WAN	BGP per overlay
Internet + MPLS (or other examples of segregated transports)	BGP per overlay



# Preparing Jinja templates

As mentioned in the [Deployment overview on page 6](#) section, we will be mostly using the Jinja Templates to configure **Underlay**, **Overlay**, and **Routing** pillars of our solution.

We provide the templates for both routing design methods discussed in [Routing design methods on page 13](#). The deployment workflow described throughout this document is mostly *agnostic* to the chosen design method: all you need is to download the right set of templates from our GitHub repository (see [Appendix B - External resources on page 66](#)), and follow the steps described in the subsequent sections.



This document focuses on the new [BGP on loopback on page 14](#) design method. We will, however, explicitly *highlight* all the steps that require adjustments, should you choose the traditional [BGP per overlay on page 13](#) design method.

This section includes the following topics:

- [Understanding template structure on page 17](#)
- [Defining the Project template on page 18](#)

## Understanding template structure

All the "mutable" parts of the template set is concentrated in the `Project` file, which we will refer to as a **Project template**. This is where you tailor the templates set to your particular project(s). The rest of the files are not expected to change between projects. They simply use the structures defined in the Project template, in order to produce the Best Practice configuration.

Here is the general structure of our Project template:

```
{# Set some global values #}
{# ... #}

{# Define the regions #}
{% set regions = {
    {# ... #}
}
%}

{# Define device profiles #}
{% set profiles = {
```

```

    {# ... #}
  }
%}

{# Define Hubs #}
{% set hubs = {
    {# ... #}
  }
%}

```

Apart from some global values, we define three important structures here:

- The `regions` dictionary defines the regions, listing the Hubs that serve each one of them.
- The `hubs` dictionary defines all those Hubs, and especially all the overlays that they create.
- The `profiles` dictionary defines device profiles, mainly their local topology, that is:
  - What interfaces are connected to the LAN side and what are connected to the WAN side
  - Where DHCP is available and where it is not
  - Which overlays are terminated on each WAN interface
  - And so on

All the other Jinja templates import the Project template and use it as a dataset.

Following is a brief description of the remaining templates:

- `Edge-Underlay` and `Hub-Underlay` configure underlay interfaces, based on the device profile. The profile will be assigned using a per-device meta field (variable) named `profile`.
- `Edge-Overlay` and `Hub-Overlay` configure IPsec overlay tunnels. The Edge will build the tunnels by way of each of its WAN interfaces and to each of the Hubs serving its region. We will use two per-device meta fields here: `region` and `profile`.
- `Edge-Routing` and `Hub-Routing` configure BGP peering. It covers BGP sessions to each of the Hubs serving the region. We use mostly the `region` meta field here.
- `Hub-MultiRegion` configures inter-regional connectivity. It covers Hub-to-Hub overlays and routing. It is kept separate for better clarity.

There are additional templates under the `pre-run` subfolder. Those are the **Pre-Run CLI Templates** for different devices. They are used for one-time modifications that must be applied to the freshly added model device for a successful provisioning. For example, they can be used to delete a certain factory-default policy that would interfere with the execution of other provisioning templates. Or they can be used to fix some "imperfections" of your environment without breaking the main workflow.

Pre-Run CLI Templates run *exactly once* on the model device, *before* any other provisioning templates.

## Defining the Project template

In order to "tune" the templates to your project, you must describe it in the Project template. As explained in [Understanding template structure on page 17](#), there is generally no need to edit any other templates.

The provided Project template contains examples and comments that you can use to adjust it to your project. Typically you will need to complete the following steps:

## DEFINING THE PROJECT TEMPLATE

1. Define the following two aggregate subnets for your project:

Parameter	Description
lan_summary	A subnet summarizing all the corporate (internal) prefixes in the project
lo_summary	A subnet summarizing all the SD-WAN device loopbacks in the project



In the [BGP per overlay on page 13](#) design method, lo\_summary is replaced by tunnel\_summary, summarizing all the tunnel subnets in the project.

2. Define the regions of the project. For each region, define the following parameters:

Parameter	Description
as	Autonomous System number for the region
lan_summary	A subnet summarizing all the corporate (internal) prefixes in the region
lo_summary	A subnet summarizing all the SD-WAN device loopbacks in the region
hubs	List of the Hubs serving the region



In the [BGP per overlay on page 13](#) design method, lo\_summary is not required.

3. Define the device profiles for the project. Each profile must list the interfaces and describe their roles and settings. The following parameters shall be defined for each interface:

Parameter	Description
name	Interface name, as it appears on the FortiGate device
role	Interface role (such as LAN-facing or WAN-facing)
ol_type	Overlay to be established over this interface
ip	IP address (including mask) or dhcp keyword for DHCP client

4. Define the Hubs serving the project. All the Hubs referenced in the regions definition must be described here. For each Hub, the following parameters shall be defined:

Parameter	Description
lo_bgp	Hub loopback IP, used for BGP termination
overlays	Dictionary describing the overlays served by this Hub

All the overlays referenced in the device profiles using ol\_type parameter must be described here. For each overlay, the following parameters shall be defined:

Parameter	Description
wan_ip	Hub underlay IP used to build this overlay
network_id	Network ID used to connect to this overlay

This overlay information will be used by the Edge devices to establish IPSEC tunnels to the Hubs.



In the [BGP per overlay on page 13](#) design method, `tunnel_net` parameter is also required to define the tunnel subnet for each overlay.

For a complete reference of the Project template, see the GitHub repository in [Appendix B - External resources on page 66](#). The above description includes only the most common parameters.

## Example project template

Here we are going to complete the above steps for our example project.

### 1. Define the LAN summary and the loopback summary:

```
{% set lo_summary = '10.200.0.0/14' %}
{% set lan_summary = '10.0.0.0/8' %}
```

Additionally, enable ADVPN across the regions:

```
{% set multireg_advpn = true %}
```

### 2. Define the two regions:

```
{% set regions = {
  'West': {
    'as': '65001',
    'lan_summary': '10.0.0.0/14',
    'lo_summary': '10.200.1.0/24',
    'hubs': [ 'site1-H1', 'site1-H2' ]
  },
  'East': {
    'as': '65002',
    'lan_summary': '10.4.0.0/14',
    'lo_summary': '10.200.2.0/24',
    'hubs': [ 'site2-H1' ]
  }
}
%}
```

### 3. Define the two device profiles:

```
{% set profiles = {
  'Silver': {
    'interfaces': [
      {
        'name': 'port1',
```

```

        'role': 'wan',
        'ol_type': 'ISP1',
        'ip': 'dhcp'
    },
    {
        'name': 'port4',
        'role': 'wan',
        'ol_type': 'MPLS',
        'ip': mpls_wan_ip
    },
    {
        'name': 'port5',
        'role': 'lan',
        'ip': lan_ip
    }
]
},
'Gold': {
    'interfaces': [
        {
            'name': 'port1',
            'role': 'wan',
            'ol_type': 'ISP1',
            'ip': 'dhcp'
        },
        {
            'name': 'port2',
            'role': 'wan',
            'ol_type': 'ISP2',
            'ip': 'dhcp'
        },
        {
            'name': 'port4',
            'role': 'wan',
            'ol_type': 'MPLS',
            'ip': mpls_wan_ip
        },
        {
            'name': 'port5',
            'role': 'lan',
            'ip': lan_ip
        }
    ]
}
}
%}

```

### Notes:

- In our example project, all the Internet links receive their connectivity information from the DHCP servers. Hence, we use the keyword `dhcp`.

- The links connecting to the MPLS network, on the other hand, do not use DHCP. Instead, their underlay IP addresses will be defined on a per-device basis, using FortiManager meta field `mpls_wan_ip`.

#### 4. Define the Hubs:

```
{% set hubs = {

    'site1-H1': {
        'lo_bgp': '10.200.1.253',
        'overlays': {
            'ISP1': {
                'wan_ip': '100.64.1.1',
                'network_id': '11'
            },
            'ISP2': {
                'wan_ip': '100.64.1.9',
                'network_id': '12'
            },
            'MPLS': {
                'wan_ip': '172.16.1.5',
                'network_id': '13'
            }
        }
    },

    'site1-H2': {
        'lo_bgp': '10.200.1.254',
        'overlays': {
            'ISP1': {
                'wan_ip': '100.64.2.1',
                'network_id': '21'
            },
            'ISP2': {
                'wan_ip': '100.64.2.9',
                'network_id': '22'
            },
            'MPLS': {
                'wan_ip': '172.16.2.5',
                'network_id': '23'
            }
        }
    },

    'site2-H1': {
        'lo_bgp': '10.200.2.253',
        'overlays': {
            'ISP1': {
                'wan_ip': '100.64.4.1',
                'network_id': '41'
            },
            'MPLS': {
                'wan_ip': '172.16.4.5',
                'network_id': '43'
            }
        }
    }
}
```

```
    }  
  }  
  
  }  
%}
```

### Notes:

- The Hub names correspond to those referred in the regions dictionary (using `hubs` lists).
- The overlay names correspond to those referred in the device profiles dictionary (using `ol_type` parameter)

Whenever you edit your Jinja templates, it is a good idea to validate the syntax. Many online services provide syntax validation, such as j2live (<https://j2live.ttl255.com/>).

Simply copy and paste the entire template to the online service, and click *Render*. The `Project template` file will not produce any output, so if you see an empty result, this means you **do not** have any syntax errors. If you have a syntax error (such as a missing closing bracket), the rendering will fail.

---

# Building a project foundation in FortiManager

In this chapter, we are going to prepare all the Provisioning Templates, Policy Packages, Groups and other common elements necessary to deploy the Secure SD-WAN Solution. These elements must be configured once per project. They will be used by all the deployed sites.

Thanks to their generic nature, they will also be largely reused between different projects, as we have already explained in the previous chapter, with an example of Jinja CLI Templates.

Although this document describes how to perform all the actions interactively (using FortiManager GUI), remember that all the actions can be fully automated with REST API calls, as we have briefly discussed in [Deployment overview on page 6](#).

The project foundation will consist of the following elements:

- [Device groups on page 24](#)
- [System templates on page 26](#)
- [Certificate templates on page 27](#)
- [Jinja CLI templates on page 28](#)
- [SD-WAN templates on page 32](#)
- [Static route templates on page 41](#)
- [Firewall policies on page 45](#)

## Device groups

In Device Manager, create a hierarchy of device groups that correspond to the different types of sites in the project.

Start from the following two device groups at the top level:

- *Edge* group for all the Edge devices
- *Hubs* group for all the Hub devices

Several generic elements are assigned directly to the device groups named *Edge* and/or *Hubs*.

For others, a more granular device group assignment will be needed. All granular device groups must be *nested* inside the parent groups named *Edge* or *Hubs*.



## DEVICE GROUPS

We recommend creating the following nested device group hierarchy:

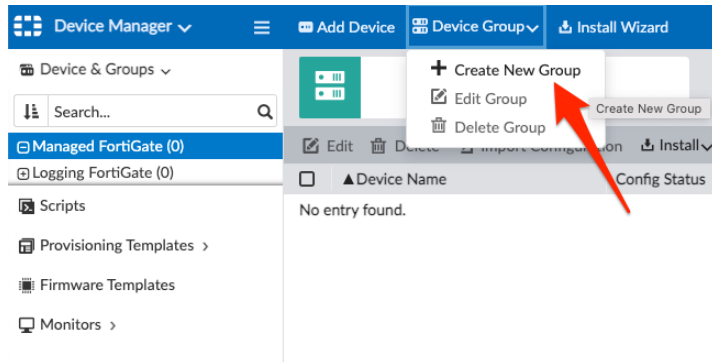
- Edge
  - Device Group for each region
    - Device Group for each device profile in the region
    - ...
  - ...
- Hubs
  - Device Group for each region
  - ...



It is logical to create a separate device group for each device profile, as shown above, because it is likely that there will be a separate SD-WAN Template assigned to it.

To create a device group:

1. In *Device Manager*, click *Device Group > Create New Group*.

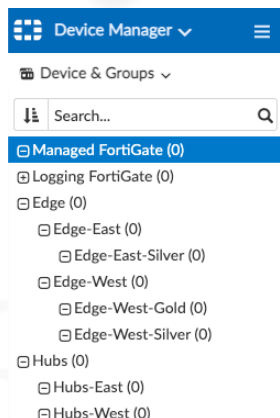


2. Enter the group name, and click *OK*. The device group is created.

Applying the above guidance to our example project, we end up with the following device groups:

- Edge
  - Edge-West
    - Edge-West-Silver
    - Edge-West-Gold
  - Edge-East
    - Edge-East-Silver
- Hubs
  - Hubs-West
  - Hubs-East

Following is an example screenshot:



## System templates

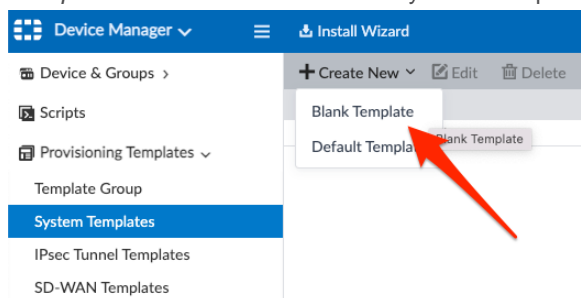
Create one or more system templates to configure basic device settings. Basic device settings typically include:

- DNS servers
- Log settings that configure devices to send logs to FortiAnalyzer
- Admin settings
- ...

After you create system templates, assign them to the corresponding device groups.

To create and assign system templates:

1. In *Device Manager*, go to *Provisioning Templates > System Templates*, and click *Create New > Blank Template* to create one or more system templates to configure basic device settings.



Here is an example of a system template named *Basic Settings*:

2. Select the template, and click *Assign to Device/Group* to assign the newly created system template(s) to the corresponding device group(s).  
For example, if you want to use a single system template for all the deployed devices, assign it directly to the parent groups named *Edge* and *Hubs*:

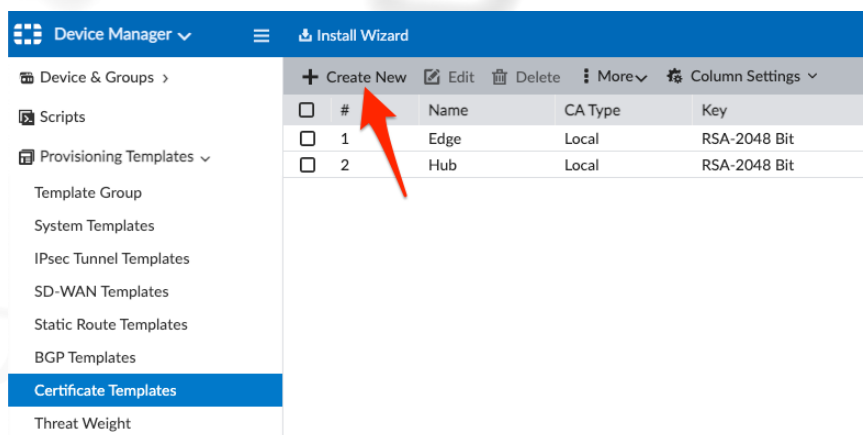
#	Name	Assigned to Device/Group
1	Basic-Settings	0 Device in Total Edge (0) Hubs (0)
2	default	0 Device in Total

## Certificate templates

The certificate templates are used to issue certificates for IPsec authentication. We will create two certificate templates named *Edge* and *Hub*.

To create a certificate templates:

1. In *Device Manager*, go to *Provisioning Templates > Certificate Templates*, and click *Create New* to create two templates named *Edge* and *Hub*:



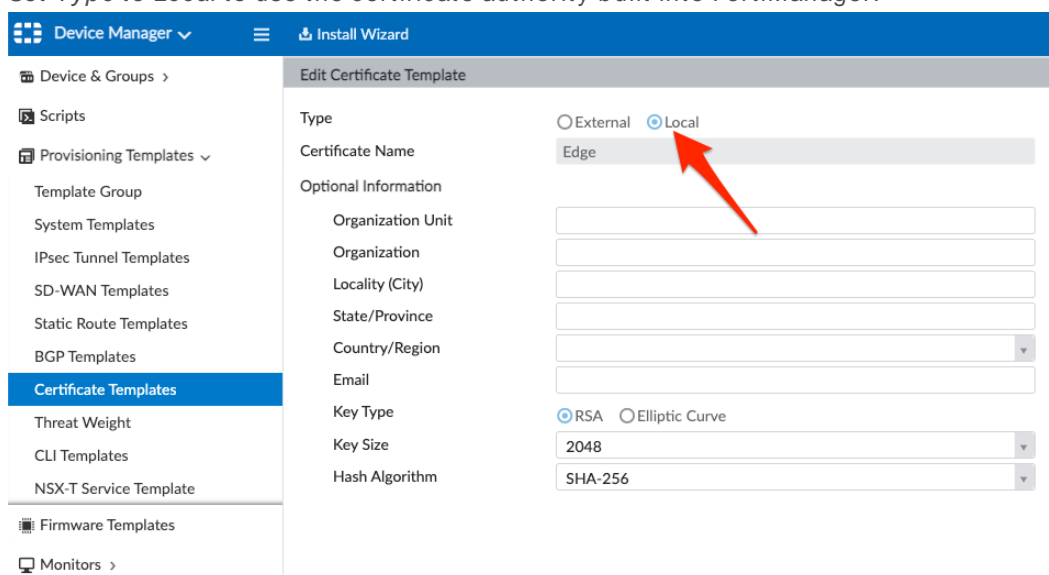
#	Name	CA Type	Key
1	Edge	Local	RSA-2048 Bit
2	Hub	Local	RSA-2048 Bit

- Set *Certificate Name* to the names used in the IPsec configuration.



The *Certificate Name* field is used for the name of the generated certificate. In our example, the certificates are named *Edge* and *Hub*. Therefore, the certificate template names must correspond to the names used in the IPsec configuration. In our case, the IPsec configuration is generated by the Jinja CLI templates that use the *Edge* and *Hub* names by default.

- Set *Type* to *Local* to use the certificate authority built into FortiManager:



Edit Certificate Template	
Type	<input type="radio"/> External <input checked="" type="radio"/> Local
Certificate Name	Edge
Optional Information	
Organization Unit	
Organization	
Locality (City)	
State/Province	
Country/Region	
Email	
Key Type	<input checked="" type="radio"/> RSA <input type="radio"/> Elliptic Curve
Key Size	2048
Hash Algorithm	SHA-256

Although this example uses the certificate authority (CA) built into FortiManager, an external CA is also supported. You can use a third-party product or FortiAuthenticator as an external CA. FortiAuthenticator is worth considering, and it can run inside FortiManager as a management extension application (MEA)!

- Configure the required certificate parameters, and save the templates.

## Jinja CLI templates

Following is a summary of the tasks to complete in Device Manager:

- Import Jinja CLI templates. See [Importing Jinja CLI templates on page 29](#).
- Create CLI template groups. See [Creating CLI template groups on page 31](#).

3. Assign CLI template groups to device groups. See [Assigning CLI template groups on page 32](#).

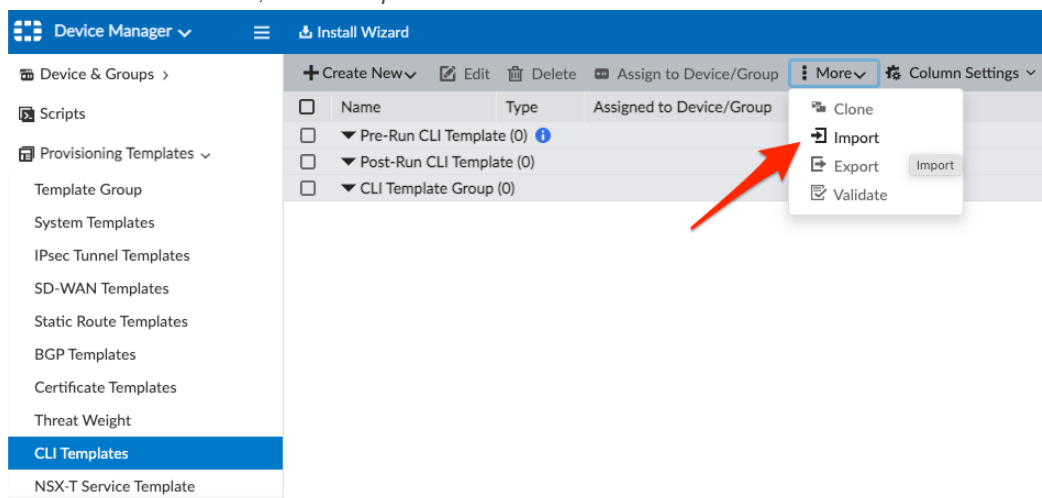
## Importing Jinja CLI templates

Import the *Project* template file first, and then import the remaining templates. The *Project* template file is imported first because all the other templates use it, so the *Project* template file must exist in FortiManager before we import the other templates.

When you import CLI templates that use meta fields, FortiManager prompts you to create the meta fields.

To import Jinja CLI templates:

1. In *Device Manager*, go to *Provisioning Templates > CLI Templates*.
2. From the *More* menu, select *Import*.

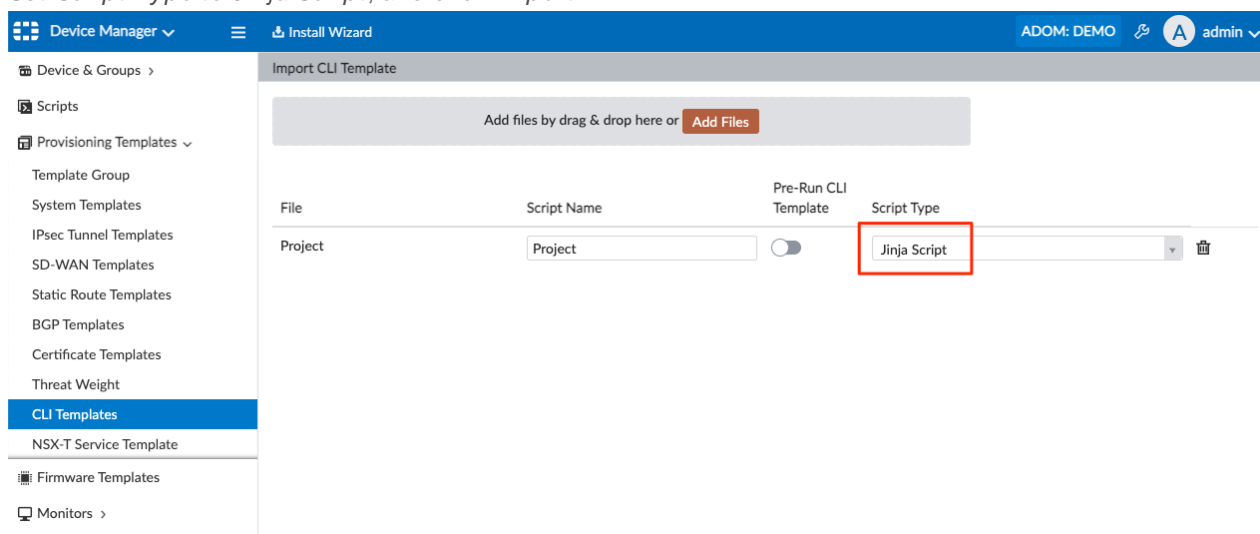


3. Drag and drop the *Project* template file on the *Import CLI Template* pane.



Ensure that you import the *Project* template file first. Because all the other templates use the *Project* template, it must exist in FortiManager before you import the other templates.

4. Set *Script Type* to *Jinja Script*, and click *Import*:



FortiManager automatically suggests that you create all the necessary meta fields.

5. Create the necessary meta fields. Ensure that you create them all as optional *Device* meta fields:

The following meta fields are missing:

- inbandwidth
- lan\_ip
- mpls\_wan\_ip
- outbandwidth
- shaping\_profile

Create Missing Meta Fields

#	Missing Meta Fields	Device	Device VDOM	Required	Length
1	inbandwidth	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	20
2	lan_ip	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	20
3	mpls_wan_ip	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	20
4	outbandwidth	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	20
5	shaping_profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	20

Create Missing Meta Fields Cancel

6. Click *Import* again. The import succeeds.
7. Import all the other templates in our set.
- Pay special attention to the pre-run CLI templates because they must be marked explicitly.
- It is also important to set *Script Type* to *Jinja Script* for all the files:

Import CLI Template

Add files by drag & drop here or [Add Files](#)

File	Script Name	Pre-Run CLI Template	Script Type
01-Edge-Underlay.j2	01-Edge-Underlay	<input type="checkbox"/>	Jinja Script
01-Hub-Underlay.j2	01-Hub-Underlay	<input type="checkbox"/>	Jinja Script
02-Edge-Overlay.j2	02-Edge-Overlay	<input type="checkbox"/>	Jinja Script
02-Hub-Overlay.j2	02-Hub-Overlay	<input type="checkbox"/>	Jinja Script
03-Edge-Routing.j2	03-Edge-Routing	<input type="checkbox"/>	Jinja Script
03-Hub-Routing.j2	03-Hub-Routing	<input type="checkbox"/>	Jinja Script
04-Hub-MultiRegion.j2	04-Hub-MultiRegion	<input type="checkbox"/>	Jinja Script
FGTVM-initial.j2	FGTVM-initial	<input checked="" type="checkbox"/>	Jinja Script

8. FortiManager will suggest creating additional meta fields (including *profile* and *region*). As before, make sure to create them all as optional *Device* meta fields:

**Missing Meta Fields**

The following meta fields are missing:

- ✗ hostname
- ✗ loopback
- ✗ profile
- ✗ region

Script Type  
Jinja Script

**Create Missing Meta Fields**

Column Settings  Search...

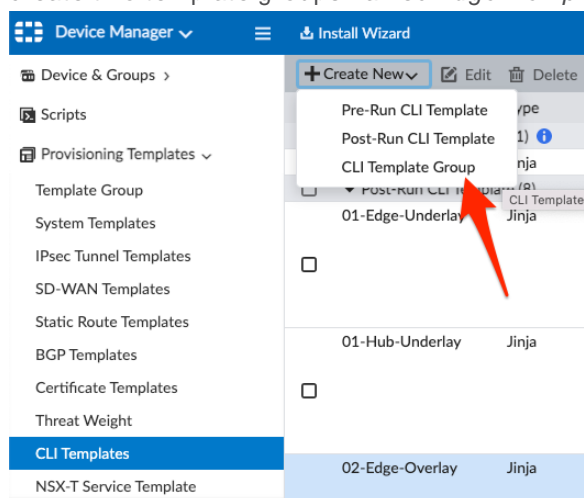
<input type="checkbox"/>	#	Missing Meta Fields	Device	Device VDOM	Required	Length
<input type="checkbox"/>	1	hostname	✓	<input type="checkbox"/>	<input type="checkbox"/>	20
<input type="checkbox"/>	2	loopback	✓	<input type="checkbox"/>	<input type="checkbox"/>	20
<input type="checkbox"/>	3	profile	✓	<input type="checkbox"/>	<input type="checkbox"/>	20
<input type="checkbox"/>	4	region	✓	<input type="checkbox"/>	<input type="checkbox"/>	20

9. Click *Import* again, and make sure the import succeeds.

## Creating CLI template groups

To create CLI template groups:

1. On the *Provisioning Templates > CLI Templates* pane, click *Create New > CLI Template Group*, and create two template groups named *Edge-Template* and *Hub-Template*:



For example, create the *Edge-Template* group, and select the following members:

- 01-Edge-Underlay
- 02-Edge-Overlay
- 03-Edge-Routing

**Edit CLI Template Group**

Template Group Name: Edge-Template

Comments:

Members:

- 01-Edge-Underlay
- 02-Edge-Overlay
- 03-Edge-Routing

For example, create the *Hub-Template* group, and select the following members

- 01-Hub-Underlay
- 02-Hub-Overlay
- 03-Hub-Routing
- 04-Hub-MultiRegion

Edit CLI Template Group

Template Group Name	Hub-Template
Comments	
Members	<input type="text"/> <input type="text"/> 01-Hub-Underlay <input type="text"/> 02-Hub-Overlay <input type="text"/> 03-Hub-Routing <input type="text"/> 04-Hub-MultiRegion

## Assigning CLI template groups

The generic CLI template groups will be applied to all types of sites.

To assign CLI template groups:

1. On the *Provisioning Templates > CLI Templates* pane, click *Assign to Device/Group* to assign the CLI template groups to the device groups named *Edge* and *Hubs* respectively:

Name	Type	Assign to Device/Group	Variables
04-Hub-MultiRegion	Jinja	0 Device in Total	inbandwidth lan_ip loopback mpls_wan_ip
Project	Jinja	0 Device in Total	inbandwidth lan_ip mpls_wan_ip outbandwidth + 1 more
▼ CLI Template Group (2)			
Edge-Template	CLI/Jinja	0 Device in Total Edge (0)	
Hub-Template	CLI/Jinja	0 Device in Total Hubs (0)	

## SD-WAN templates

While the generic Jinja CLI templates mostly handle the **Underlay**, **Overlay**, and **Routing** pillars, the **SD-WAN** pillar uses the GUI-based SD-WAN templates. Furthermore, we will create a separate template for each profile and for each number of Hubs in the region.





When a site has an extra WAN link, the extra WAN link must be configured. Similarly, when a region has two Hubs, rather than one Hub, the second Hub must be configured too.

We are going to configure all the WAN-facing interfaces—both underlay and overlay—to be part of our SD-WAN bundle. This way, all the traffic steering will be controlled by the SD-WAN rules. And since our "foundation" is generic, we do not need to reconfigure the other pillars (for example, IPsec and BGP), when changing SD-WAN rules.

This section contains the following topics:

- [Creating Edge SD-WAN templates on page 33](#)
- [Creating Hub SD-WAN templates on page 39](#)

## Creating Edge SD-WAN templates

This section outlines the general configuration steps recommended for every Edge SD-WAN template. All the screenshots in this section demonstrate a template named *Edge-2H-Silver* from our example project, and the template is prepared for the sites from the West Region that have a Silver profile (such as *site1-1*). See [Example project on page 11](#).



Detailed description of the available SD-WAN functionality is outside the scope of this document. Please refer to the public documentation or contact your Fortinet representatives for more details.

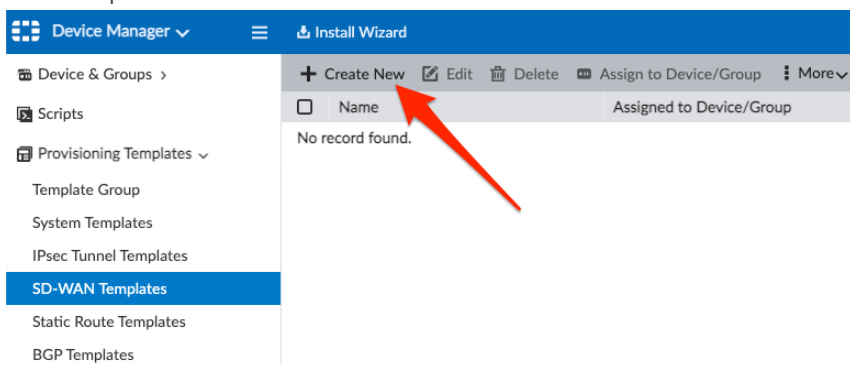
## Creating SD-WAN templates

The section provides the general steps for creating the following components of SD-WAN templates:

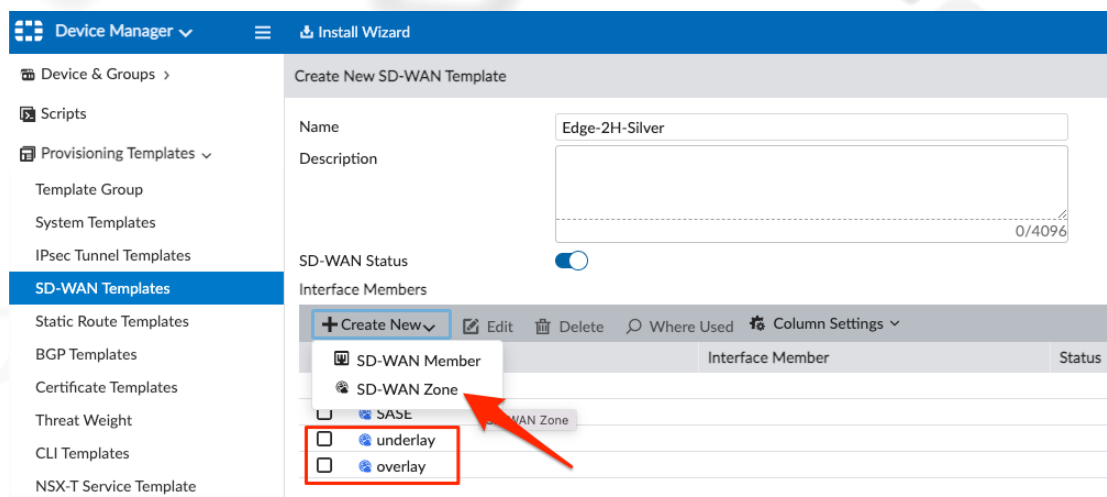
- SD-WAN zones
- SD-WAN zone members
- Performance SLAs
- SD-WAN rules for steering traffic

To create SD-WAN templates:

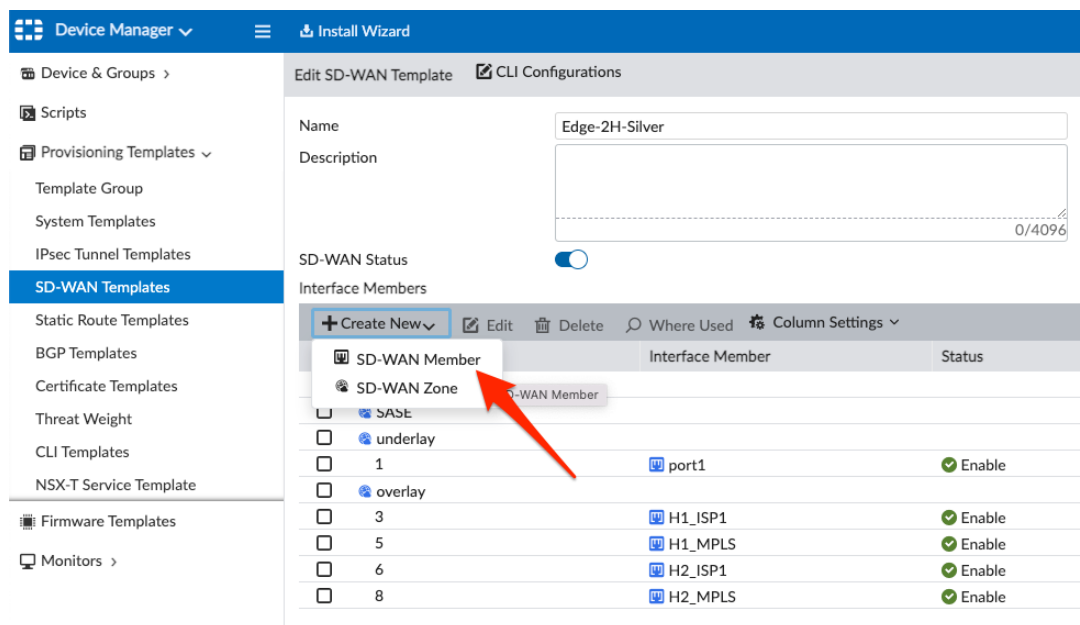
1. In *Device Manager*, go to *Provisioning Templates > SD-WAN Templates*, and click *Create New* to create a new template:



2. In the *Interface Members* section, click *Create New > SD-WAN Zone*, and create two SD-WAN zones named *underlay* and *overlay*:



3. In the *Interface Members* section, click *Create New* > *SD-WAN Member* to create SD-WAN Members, and add them to the zones:



- Add all the underlay interfaces to the *underlay* zone.
- Add all the overlay interfaces (tunnels) to the *overlay* zone.
- Make sure that the *Priority* value for the overlay members is worse (higher) than the value for the underlay members. For example, set the overlay *Priority* to 10 and leave the underlay *Priority* unchanged (0).



We are going to use SD-WAN as a default route, and this priority will be set for each individual members. As a result, underlay members will be automatically preferred for Internet traffic that does not have any explicit SD-WAN rule. This also includes the traffic originated by the FortiGate device itself, such as FortiGuard connectivity, for example.

- Make sure that the *Source* value for the overlay members is set to  $\$(loopback)$ . This refers to the per-device meta field that will contain the per-device loopback address, and in our routing design flavor we will use it as a source IP for the health probes.

## Edit SD-WAN Interface Member

Sequence Number	3
Interface Member	H1_ISP1
SD-WAN Zone	overlay
Gateway IP	0.0.0.0
Cost	0
Status	<input checked="" type="checkbox"/>
Priority	10
Weight	1
Advanced Options	<div>comment</div> <div>gateway6 ::</div> <div>priority6 1024</div> <div>source \$(loopback)</div> <div>source6 ::</div>

OK

Cancel

4. In the *Performance SLA* section, click *Create New* to create the necessary Performance SLAs:

- For the corporate (internal) traffic, create Performance SLA pinging the Hub loopback. This loopback is created by the Jinja CLI Templates, and its IP address, by default, is set to 10.200.99.1. Specify to run the probes only on the overlay members.

## Edit Performance SLA

Name	HUB		
IP Version	IPv4	IPv6	
Probe Mode	Active	Passive	Prefer Passive
Protocol	Ping	TCP ECHO	UDP ECHO HTTP TWAMP DNS TCP CONNECT FTP
Server	10.200.99.1		
Participants	<div>All SD-WAN Members Specify</div> <div> <div>H1_ISP1</div> <div>H1_MPLS</div> <div>H2_ISP1</div> <div>H2_MPLS</div> </div> <div>4 Entries Selected</div>		
Enable Probe Packets	<input checked="" type="checkbox"/>		
SLA Targets	<div>Target 1</div> <div> <div>Latency Threshold</div> <div>Jitter Threshold</div> <div>Packet Loss Threshold</div> </div> <div> <div>100</div> <div>Milliseconds</div> </div> <div>+ Add Target</div>		

- For the Internet traffic, create Performance SLAs using any desired probe destination. A typical choice to probe generic Internet connectivity would be a DNS probe towards 8.8.8.8. Additionally, application-specific Performance SLAs can be created, depending on the desired steering strategy. Specify to run the probes through all the members that should be used for Internet access. The choice depends on whether it is required to provide Direct Internet Access (DIA), Remote Internet Access (RIA), or a hybrid of the two.

For example, in the following image we are planning to use a hybrid Internet access model. However, we do not define any traffic steering rules yet. We are merely defining on which SD-WAN Members to run the respective health probes. The traffic steering will be determined by the SD-WAN Rules.



If you want to see SLA graphs on the FortiAnalyzer widgets and reports, remember to set the values of `sla-fail-log-period` and `sla-pass-log-period` under *Advanced Options* of the respective Performance SLAs! For example, use the values of 10 seconds as displayed in the following image:

5. In the *SD-WAN Rules* section, click *Create New* to create the SD-WAN rules that will steer traffic:

- For the corporate (internal) traffic, use the following guidelines for optimal ADVPN operation:
  - Enable an Advanced Option *tie-break fib-best-match*. This per-rule option instructs SD-WAN to rely on the best route to the destination (rather than on *any* valid route, as it is done by default). In conjunction with ADVPN, this setting provides an optimal behavior in certain failure scenarios.

- When *Lowest Cost (SLA)* strategy is used, set an Advanced Option *hold-down-timer* to 20 seconds. In conjunction with ADVPN, this setting prevents unnecessary traffic flapping in certain failure or recovery scenarios.

## Edit SD-WAN Rule

gateway	<input type="checkbox"/>
hash-mode	round-robin
hold-down-time	20
input-device	<input type="text" value="Click to select"/>
input-device-negate	<input type="checkbox"/>
jitter-weight	0
latency-weight	0
link-cost-threshold	10
minimum-sla-meet-members	0
packet-loss-weight	0
passive-measurement	<input type="checkbox"/>
role	standalone
sla-compare-method	order
src-negate	<input type="checkbox"/>
standalone-action	<input type="checkbox"/>
status	<input checked="" type="checkbox"/>
tie-break	fib-best-match
use-shortcut-sla	<input checked="" type="checkbox"/>

- In Dual-Hub regions, we recommend using an "Active/Backup Hub" model. This means creating two separate SD-WAN Rules, such as *Corporate-H1* and *Corporate-H2*, with the former listing only the overlays of the Primary Hub and the latter listing only the overlays of the Secondary Hub. As long as the Primary Hub is operational, it will be used for ADVPN shortcut exchange. Once it becomes out of service, the Secondary Hub will be used.

Device Manager | Install Wizard

Device & Groups | Scripts | Provisioning Templates | SD-WAN Templates | Static Route Templates | BGP Templates | Certificate Templates | Threat Weight | CLI Templates | NSX-T Service Template | Firmware Templates | Monitors

Edit SD-WAN Template | CLI Configurations

Performance SLA

Name	Health-Check Server	Detect Protocol
HUB	10.200.99.1	Ping
Internet	8.8.8.8	DNS

SD-WAN Rules

ID	Name	Source	Destination	Criteria	Members
1	Corporate-H1	CORP_LAN	CORP_LAN	SLA (HUB#1)	H1_ISP1 H1_MPLS
2	Corporate-H2	CORP_LAN	CORP_LAN	SLA (HUB#1)	H2_ISP1 H2_MPLS
3	Business-Critical-SaaS	ALL	Salesforce GoToMeeting	SLA (Internet#1)	port1 H1_MPLS H2_MPLS
	sd-wan	ALL	ALL	Sessions	ALL

- For the Internet traffic, a wide variety of options exists, depending on the desired steering strategy. For example, the following screenshot demonstrates an SD-WAN Rule for the business-critical SaaS applications ("Salesforce" and "GoToMeeting" in our example), implementing a hybrid Internet access model combined with an Application-based traffic steering:

## Edit SD-WAN Rule

**Name** Business-Critical-SaaS

**IP Version** IPv4

**Source**

Source Address Click here to select

Users Click here to select

User Groups Click here to select

**Destination**

Address **Internet Service**

Internet Service Click here to select

Internet Service Group Click here to select

Custom Internet Service Click here to select

Custom Internet Service Group Click here to select

**Application**

Search

Salesforce id: 16920

GoToMeeting id: 16354

2 Entries Selected

**Application Group**

Click here to select

**Type of Service**

0x00 Bit Mask 0x00

**Outgoing Interfaces**

Strategy Manual Best Quality **Lowest Cost (SLA)** Maximize Bandwidth (SLA)

**Interface Preference**

Search

port1

H1\_MPLS

H2\_MPLS

3 Entries Selected

**Zone Preference**

Click here to select

**Required SLA Target**

Search

Internet#1  
DNS, 8.8.8.8; Latency: 250ms, Jitter: 5ms, Packet Loss: 0%

1 Entry Selected

Advanced Options &gt;

OK

Cancel

- The applications in question will be identified by their Layer7 payloads, using our extensive Application database
- The applications will prefer Direct Internet Access (using "port1"), as long as it meets the configured SLA target (the latency of up to 250 ms).
- If the SLA target is not met, the traffic will switchover to the Remote Internet Access via one of the Hubs (using one of the MPLS overlays - H1\_MPLS or H2\_MPLS).
- Different SD-WAN Rules can be created for different types of traffic, selecting the optimal traffic steering strategy and SLA targets for different applications.

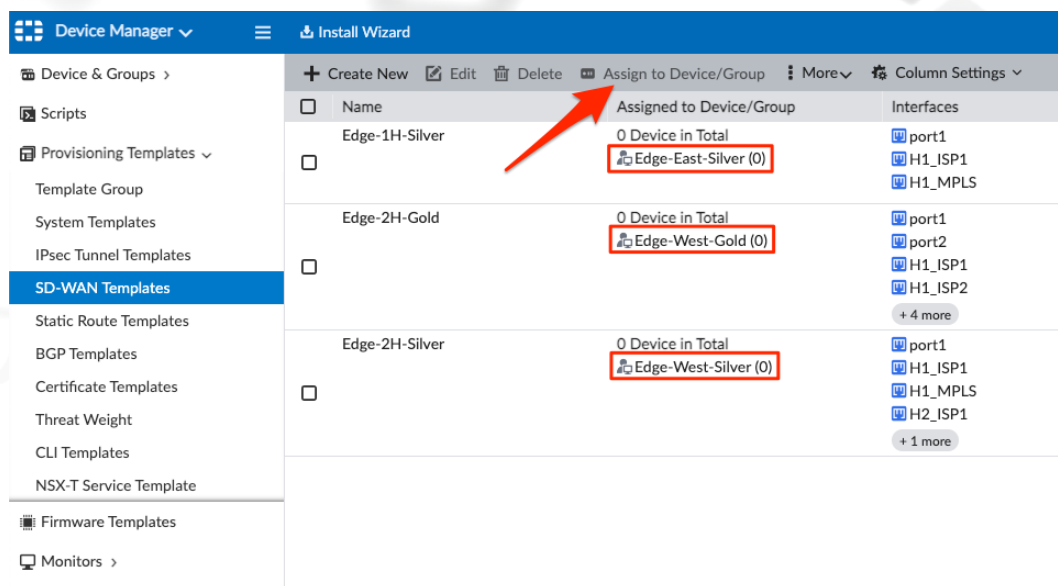
## Assigning SD-WAN templates

Once the SD-WAN templates are ready, assign them to the correct device groups. The outcome will vary depending on how you defined the device groups for your project.

To assign SD-WAN templates:

1. On the *Provisioning Templates > SD-WAN Templates* pane, select the template, and click *Assign to Device/Group* to assign the templates to the correct device groups. You can assign the same SD-WAN template to one or more device groups.

The following image demonstrates the assignment in our example project:



## Creating Hub SD-WAN templates

For the consistent configuration, we also recommend enabling SD-WAN on the Hubs. When creating SD-WAN templates for the Hub, follow general steps 1-3 that are described in [Creating Edge SD-WAN templates on page 33](#).

This includes creating the two SD-WAN zones named *underlay* and *overlay*.



On the Hubs, the members of the *overlay* zone will be the dial-up tunnel interfaces.

The image below demonstrates a template named *Hub-Gold* from our example project, and the template is prepared for Hub devices that serve the West Region (such as *site1-H1*):

**Device Manager** ▾ **Install Wizard**

Device & Groups ▸

Scripts

Provisioning Templates ▾

Template Group

System Templates

IPsec Tunnel Templates

**SD-WAN Templates**

Static Route Templates

BGP Templates

Certificate Templates

Threat Weight

CLI Templates

NSX-T Service Template

Firmware Templates

Monitors ▸

**Edit SD-WAN Template** **CLI Configurations**

Name: Hub-Gold

Description:   
0/4096

SD-WAN Status: ☒

Interface Members

ID	Interface Member	Status	Weight
<input type="checkbox"/>	virtual-wan-link		
<input type="checkbox"/>	SASE		
<input type="checkbox"/>	underlay		
<input type="checkbox"/>	1 port1	Enable	1
<input type="checkbox"/>	2 port2	Enable	1
<input type="checkbox"/>	overlay		
<input type="checkbox"/>	3 EDGE_ISP1	Enable	1
<input type="checkbox"/>	4 EDGE_ISP2	Enable	1
<input type="checkbox"/>	5 EDGE_MPLS	Enable	1

Performance SLA

Name	Health-Check Server	Detect Protocol
Internet	8.8.8.8	DNS

Regarding the steps 4-5 (Performance SLAs and SD-WAN Rules) from [Creating Edge SD-WAN templates on page 33](#), the following guidelines apply:

- Generally, configuring Performance SLAs and/or SD-WAN Rules on the Hubs is not mandatory. Quite often the Hub can provide Internet access using conventional routing, both for the workloads behind the Hub itself and for the Remote Internet Access from the Edges.
- For Edge-to-Edge traffic and ADVPN exchanges, no SD-WAN configuration is necessary on the Hubs. The Edge devices make the steering decisions for this traffic. The Hubs implement the "overlay stickiness" principle, preferring to stay within the overlay chosen by the originating Edge. This principle is implemented using Policy Routes, generated by our [Jinja CLI templates on page 28](#).
- However, if the Hub has multiple Internet access options (such as multiple ISPs), it is also possible to control the steering using SD-WAN Rules, similar to the Edges.
- Additionally, it may be beneficial to monitor generic Internet connectivity from the Hubs (for example, using DNS probes towards 8.8.8.8), if only for visibility. In this case, it is enough to configure the corresponding Performance SLAs, without any SD-WAN Rules.
- Finally, there are special cases that require SD-WAN Rules on the Hubs, such as controlling Hub-to-Edge traffic.

Once the SD-WAN Templates are ready, assign them to the correct device groups. The following image demonstrates the assignment in our example project:



Device Manager ▾			
Install Wizard			
<a href="#">+ Create New</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Assign to Device/Group</a> <a href="#">More ▾</a> <a href="#">Column Settings ▾</a>			
Device & Groups >	Name	Assigned to Device/Group	Interfaces
Scripts	Edge-1H-Silver	0 Device in Total Edge-East-Silver (0)	port1 H1_ISP1 H1_MPLS
Provisioning Templates ▾	Edge-2H-Gold	0 Device in Total Edge-West-Gold (0)	port1 port2 H1_ISP1 H1_ISP2 + 4 more
Template Group	Edge-2H-Silver	0 Device in Total Edge-West-Silver (0)	port1 H1_ISP1 H1_MPLS H2_ISP1 + 1 more
System Templates	Hub-Gold	0 Device in Total Hubs-West (0)	port1 port2 EDGE_ISP1 EDGE_ISP2 + 1 more
IPsec Tunnel Templates	Hub-Silver	0 Device in Total Hubs-East (0)	port1 EDGE_ISP1 EDGE_MPLS
SD-WAN Templates			
Static Route Templates			
BGP Templates			
Certificate Templates			
Threat Weight			
CLI Templates			
NSX-T Service Template			
Firmware Templates			
Monitors >			

## Static route templates

We recommend using a Static Route Template to configure **the default route** through SD-WAN on all devices. We will also use the static route template and meta field to configure a static route towards the entire MPLS underlay transport network.

Following is a summary of the tasks to complete in Device Manager:

1. Create a static route template. See [Creating a static route template on page 41](#).
2. Edit the static route template to add a default route. See [Adding a static default route on page 42](#).
3. Assign the static route template to the Edge and Hub device groups. See [Assigning the static route template on page 42](#).
4. Edit the static route template to add a static route towards the entire MPLS underlay transport network. See [Adding a static route for the next-hop gateway on page 43](#).

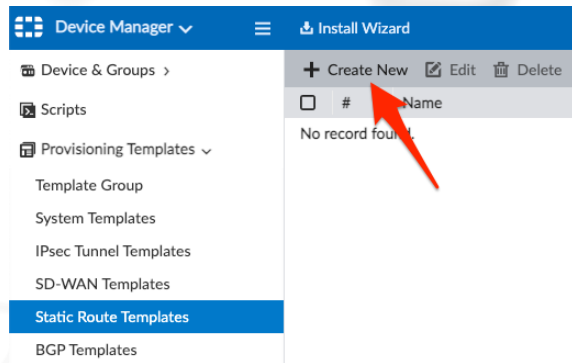
## Creating a static route template

Create a static route template. After you create the template, you will add static routes to the template.

To create a static route template:

1. In *Device Manager*, go to *Provisioning Templates > Static Route Templates*, and click *Create New* to create a new template:

## STATIC ROUTE TEMPLATES



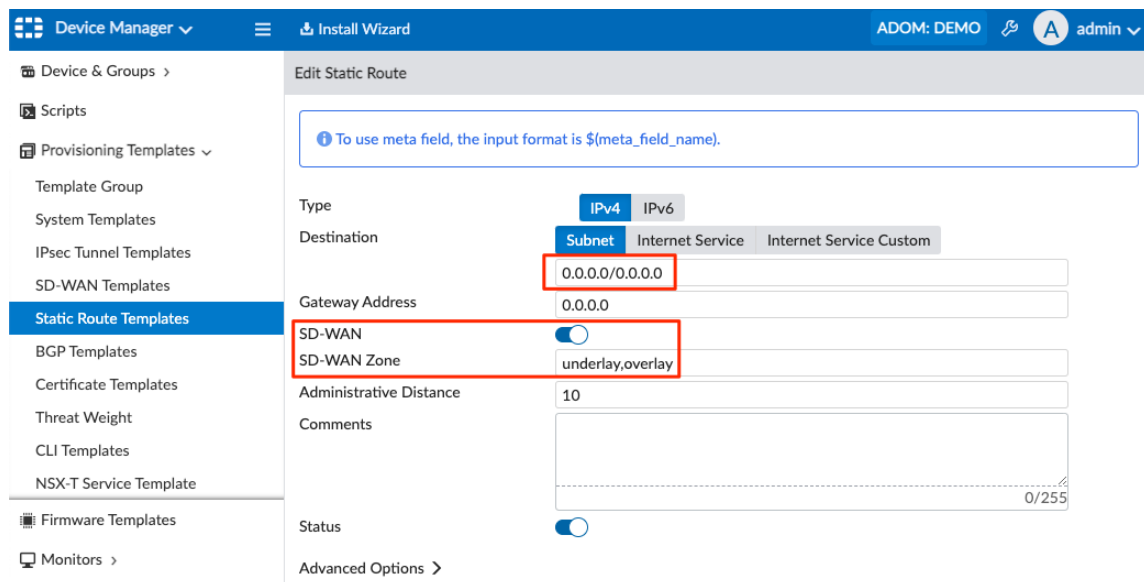
2. Type a name for the static route template, and click **OK**. The static route template is created and displayed in the content pane, and you can add static routes to the template.

## Adding a static default route

After you create a static route template, you can edit the template to add a new default route (0.0.0.0/0).

To add a default route to the static route template:

1. On the *Provisioning Templates > Static Route Templates* pane, double-click the static route template to open it for editing.
2. Click **Create New**, and set *Type* to **IPv4**.
3. Set *Destination* to **Subnet** and **0.0.0.0/0.0.0.0**.
4. Enable **SD-WAN**, and specify the two SD-WAN zones separated by a comma (without space): **underlay,overlay**.



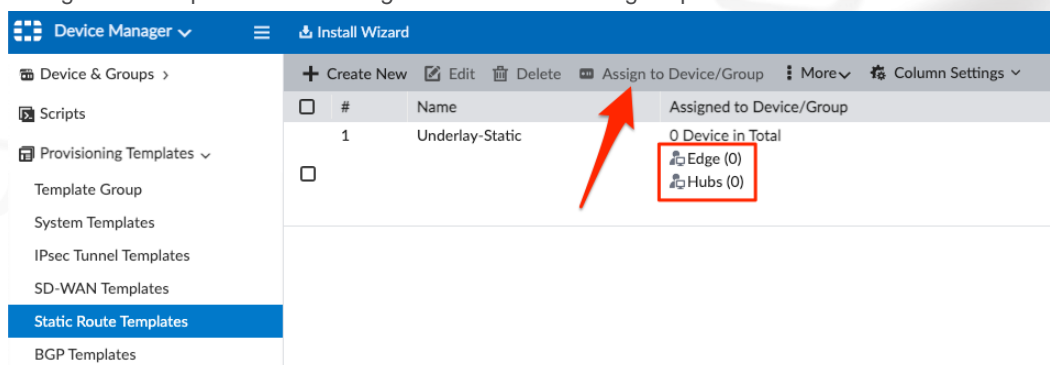
5. Click **OK** to save the static route.

## Assigning the static route template

Assign the static route template to both the Edge and Hub device groups.

To assign the static route template:

1. On the *Provisioning Templates > Static Route Templates* pane, select the static route template, and click *Assign to Device/Group*.
2. Assign the template to both *Edge* and *Hubs* device groups:



## Adding a static route for the next-hop gateway

In addition to the default route, you can use Static Route Templates for any static routes that may be necessary in your environment.

In our example project, we need to specify a next-hop gateway for the MPLS underlay transport, since it does not have a DHCP server. On every site, we will add a static route towards the entire MPLS underlay network through the right next-hop gateway.

Because the next-hop gateway is site-specific, it will be different on each site. When we were discussing the Jinja CLI Templates, we saw how to deal with site-specific information by using a per-device meta field (which is a variable). Meta fields are used to define the IP address of the MPLS underlay interface on each FortiGate device by using the `mpls_wan_ip` meta field. For details, see [Example project template on page 20](#).

We will use a similar approach for the next-hop gateway because static route templates support meta fields.

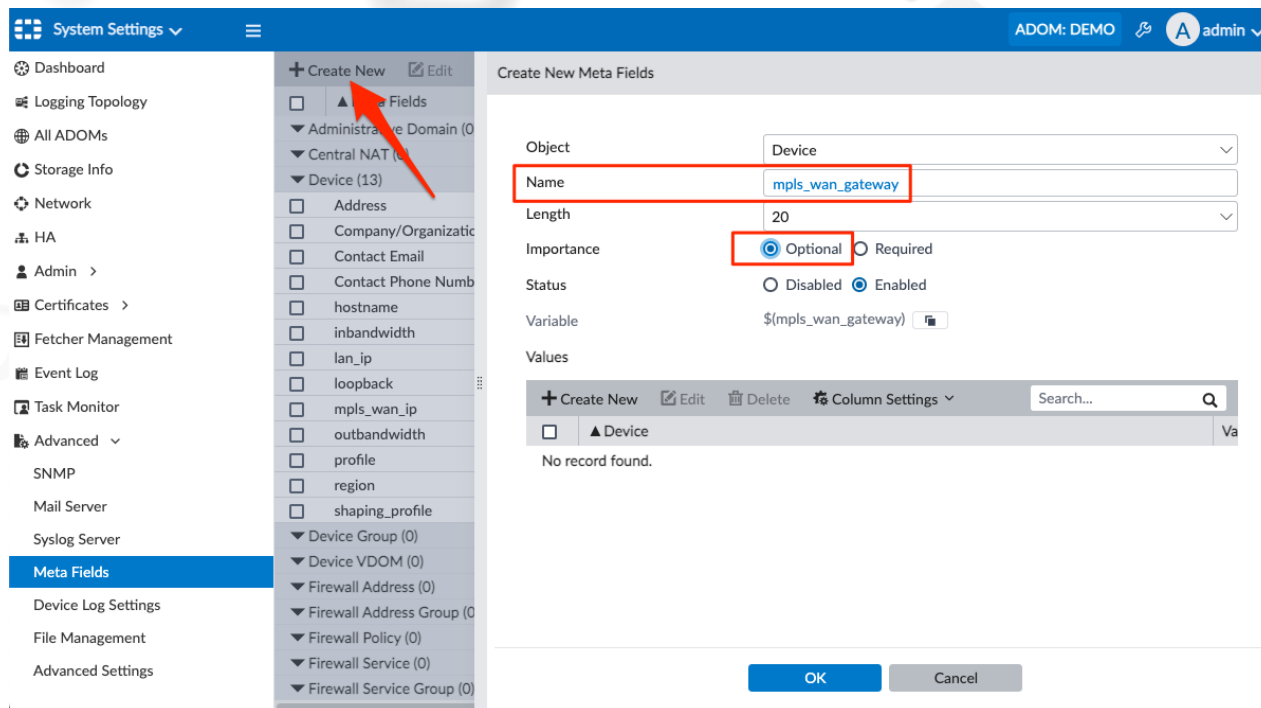
Following is a summary of the tasks to complete for our example project:

1. Create a meta field for the MPLS gateway. See [Creating a meta field on page 43](#).
2. Edit the static route template to add another route. See [Adding a static route on page 44](#).

## Creating a meta field

To create a meta field for the MPLS gateway:

1. In *System Settings*, go to *Advanced > Meta Fields*, and create a new optional Device meta field called `mpls_wan_gateway`:



The screenshot shows the 'System Settings' interface with the 'Meta Fields' section selected in the left sidebar. The 'Create New Meta Fields' dialog is open, showing the following configuration:

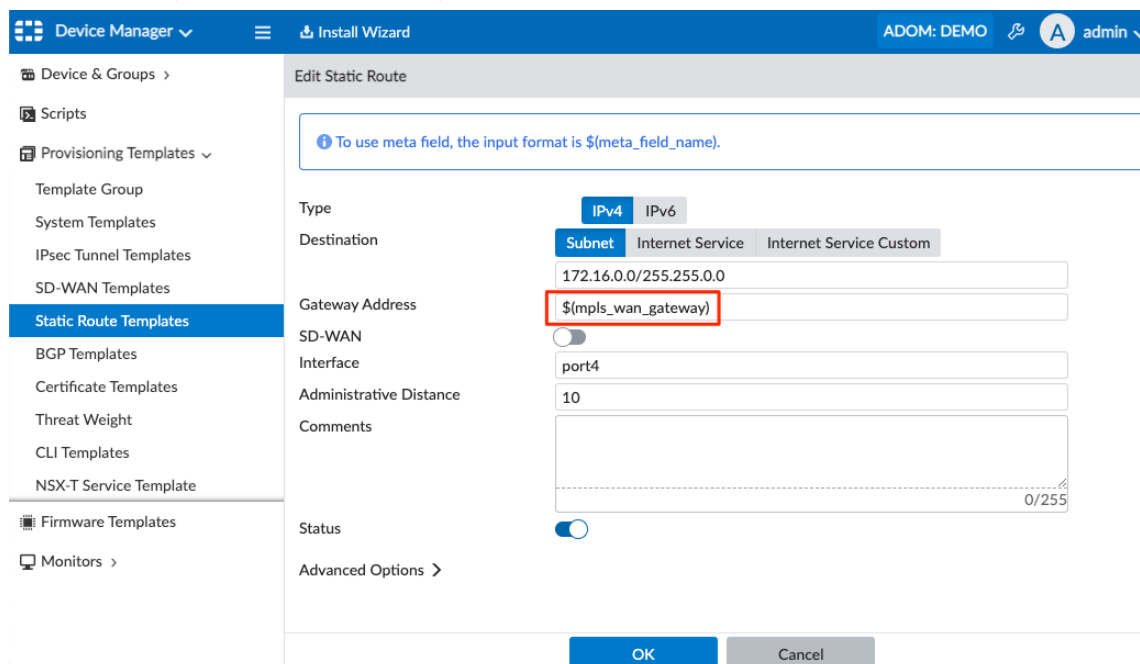
- Object:** Device
- Name:** mpls\_wan\_gateway
- Length:** 20
- Importance:** ☒ Optional ☐ Required
- Status:** ☐ Disabled ☒ Enabled
- Variable:** \$(mpls\_wan\_gateway)
- Values:** (Empty list)

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

## Adding a static route

To edit the static route template and add another route:

1. In *Device Manager*, on the *Provisioning Templates > Static Route Templates* pane, double-click the static route template to open it for editing.
2. Click *Create New*, and set *Type* to *IPv4*.
3. Set *Destination* to *Subnet*.
4. In the *Gateway Address* field specify the meta field created at the previous step:



The screenshot shows the 'Device Manager' interface with the 'Static Route Templates' section selected in the left sidebar. The 'Edit Static Route' dialog is open, showing the following configuration:

- Type:** IPv4
- Destination:** Subnet
- Destination Address:** 172.16.0.0/255.255.0.0
- Gateway Address:** \$(mpls\_wan\_gateway)
- SD-WAN:** ☐
- Interface:** port4
- Administrative Distance:** 10
- Comments:** (Empty text area)
- Status:** ☒

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

5. Click *OK* to save the static route.

## Firewall policies

For the last remaining pillar of **Security** we will be using the most standard approach, the Policy Packages.

This section contains the following topics:

- [Creating common elements on page 45](#)
- [Creating the Edge policy package on page 46](#)
- [Creating the Hub policy package on page 48](#)

### Creating common elements

First, we must create the following common elements:

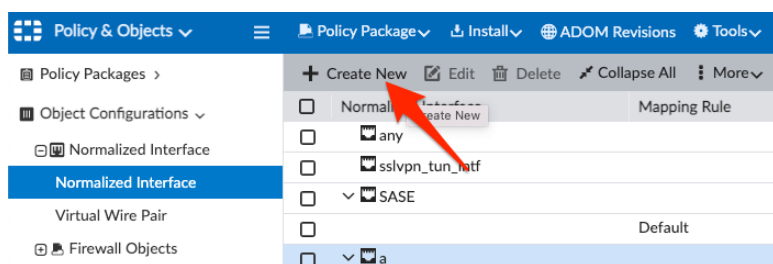
1. Create normalized interfaces. See [Creating normalized interfaces on page 45](#).
2. Create address objects. See [Creating address objects on page 45](#).

### Creating normalized interfaces

To create normalized interfaces:

1. In *Policy & Objects*, go to *Object Configurations > Normalized Interface*.
2. Click *Create New* to create the following normalized interfaces:

Name	Description
lan_zone	System Zone combining all the LAN-facing interfaces
hub2hub_overlay	System Zone combining all the Hub-to-Hub overlays (interconnecting regions)
Lo-HC	Loopback (on Hubs) used for health-check probes
Lo	Main Loopback

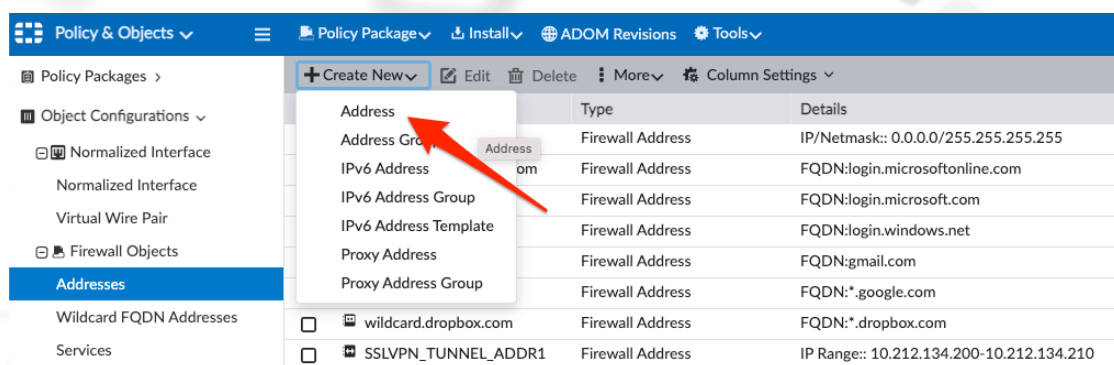


### Creating address objects

To create address objects:

1. In *Policy & Objects*, go to *Object Configurations > Firewall Objects > Addresses*.
2. Click *Create New > Address* in order to create the following object:

Name	Description
CORP_LAN	Corporate (internal) LAN summary, such as 10.0.0.0/8



Address	Type	Details
Address Group	Firewall Address	IP/Netmask:: 0.0.0.0/255.255.255.255
IPv6 Address	Firewall Address	FQDN:login.microsoftonline.com
IPv6 Address Group	Firewall Address	FQDN:login.microsoft.com
IPv6 Address Template	Firewall Address	FQDN:login.windows.net
Proxy Address	Firewall Address	FQDN:google.com
Proxy Address Group	Firewall Address	FQDN:*.google.com
<input type="checkbox"/> wildcard.dropbox.com	Firewall Address	FQDN:*.dropbox.com
<input type="checkbox"/> SSLVPN_TUNNEL_ADDR1	Firewall Address	IP Range:: 10.212.134.200-10.212.134.210



You may have already created this address object earlier, when configuring the SD-WAN Rules.

## Creating the Edge policy package

Complete the following tasks to configure the policy package for Edge devices:

1. Create an policy package named *Edge*, and assign it to the Edge device group. See [Creating an Edge policy package on page 46](#).
2. Create firewall policy rules for the Edge policy package. See [Creating firewall policy rules on page 47](#).

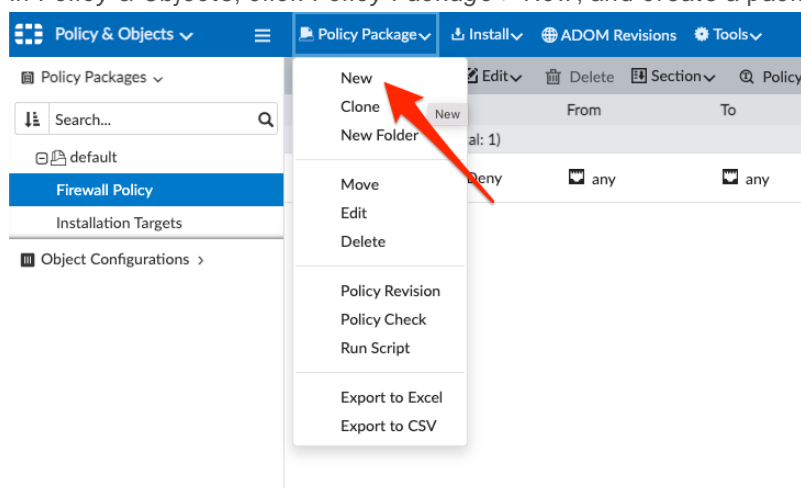
See also [Notes about the Edge policy package on page 47](#).

## Creating an Edge policy package

Create a policy package named *Edge*, and assign the policy package to the Edge device group.

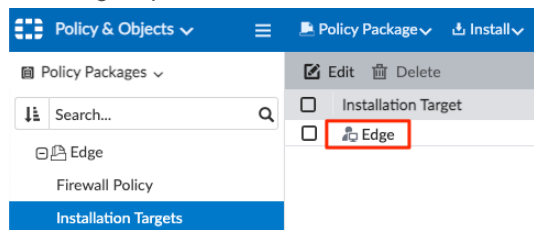
To create an Edge policy package:

1. In *Policy & Objects*, click *Policy Package* > *New*, and create a package named *Edge*:



2. Select *Policy Packages* > *Edge* > *Installation Targets*, and click *Edit* to assign the package to the *Edge*

device group:



## Creating firewall policy rules

Create firewall policy rules for the Edge policy package.

To create firewall policy rules:

1. Go to **Select Policy Packages > Edge > Firewall Policy**.
2. Click **Create New** to create the following firewall policy rules. All the rules should have **Action** set to **Accept**:

Name	From	To	Src	Dst	Service	NAT
Corporate	lan_zone overlay	lan_zone overlay	CORP_LAN	CORP_LAN	ALL	No
Internet (DIA)	lan_zone	underlay	all	all	ALL	Yes
Internet (RIA)	lan_zone	overlay	all	all	ALL	No
Health-Check	overlay	Lo	all	all	PING	No

#	Name	From	To	Source	Destination	Schedule	Service	Action	NAT	Security Profiles	Log
1	Corporate	lan_zone overlay	lan_zone overlay	CORP_LAN	CORP_LAN	always	ALL	Accept	Disabled	default certificate-inspect	Log All Sessions
2	Internet (DIA)	lan_zone	underlay	all	all	always	ALL	Accept	Enabled	default certificate-inspect	Log All Sessions
3	Internet (RIA)	lan_zone	overlay	all	all	always	ALL	Accept	Disabled	default certificate-inspect	Log All Sessions
4	Health-Check	overlay	Lo	all	all	always	PING	Accept	Disabled	no-inspection	Log Security Events
5	Implicit Deny (5-5 / Total: 1)	any	any	all	all	always	ALL	Deny			No Log

## Notes about the Edge policy package

- The Normalized Interfaces for SD-WAN zones named *underlay* and *overlay* were automatically created when we configured SD-WAN templates.
- The Normalized Interface for the LAN Zone (*lan\_zone*) was created by us manually, and our Jinja CLI Template (*Edge-Underlay*) will create the corresponding System Zone on the Edge devices.
- The Firewall Policies distinguish between Direct Internet Access (DIA, from the Edge itself) and Remote Internet Access (RIA, through the Hub), potentially applying different security features in each case. One common example is Source NAT which is only applied to the traffic using DIA.
- In the [BGP on loopback on page 14](#) design method, Edge devices will probe each other's loopback interfaces (for ADVPN Shortcut Monitoring feature). This must be explicitly permitted by Firewall Policies, as we do in the *Health-Check* rule.



In the [BGP per overlay on page 13](#) design method, this rule is not required.

- It is highly recommended enabling Application Control, especially on the Firewall Policy controlling Internet traffic. For accurate application identification, it is also highly recommended to enable SSL Inspection for security reasons, but also for the SD-WAN functionality. Remember that Application Control is required for SD-WAN application-aware traffic steering and is also used to populate SD-WAN widgets and reports.

## Creating the Hub policy package

Complete the following tasks to configure the policy package for Hub devices:

- Create an policy package named *Hub*, and assign it to the Hubs device group. See [Creating a Hub policy package on page 48](#).
- Create firewall policy rules for the Hub policy package. See [Creating firewall policy rules on page 48](#).

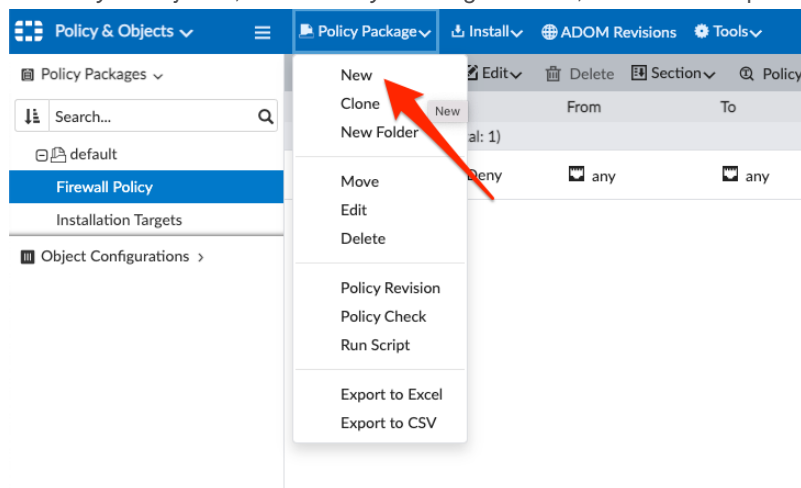
See also [Notes about the Hub policy package on page 50](#).

## Creating a Hub policy package

Create a policy package named *Hub*, and assign the policy package to the Hubs device group.

To create an Edge policy package:

- In *Policy & Objects*, click *Policy Package > New*, and create a package named *Hubs*:



- Select *Policy Packages > Hub > Installation Targets*, and click *Edit* to assign the package to the *Hubs* device group.

## Creating firewall policy rules

Create firewall policy rules for the Hub policy package.



To create firewall policy rules:

1. Go to *Select Policy Packages > Hub > Firewall Policy*.
2. Click *Create New* to create the following firewall policy rules. All the rules should have *Action* set to *Accept*:

Name	From	To	Src	Dst	Service	NAT
Edge-Edge	overlay hub2hub_ overlay	overlay hub2hub_ overlay	CORP_LAN	CORP_LAN	ALL	No
Edge-Hub	lan_zone overlay hub2hub_ overlay	lan_zone overlay hub2hub_ overlay	CORP_LAN	CORP_LAN	ALL	No
Internet (DIA)	lan_zone	underlay	all	all	ALL	Yes
Internet (RIA)	overlay	underlay	all	all	ALL	Yes
Health-Check	overlay	Lo-HC	all	all	PING	No
Peering	overlay hub2hub_ overlay	Lo	all	all	PING BGP	No

3. In the *Edge-Edge* rule, configure the following *Advanced Options*:

Parameter	Value
anti-replay	off
tcp-session-without-syn	all



Keep in mind that the Edge devices will secure the Edge-to-Edge traffic. Hence, there is no need to repeat the same inspection on the Hub. Especially considering that most of the Edge-to-Edge traffic will not even transit the Hub—it will use direct ADVPN shortcuts instead!

Furthermore, if network conditions change, the traffic can switch to another overlay and reach the Hub in the middle of the TCP session. In order to avoid traffic drop in this situation, the above *Advanced Options* are necessary. The advanced options do not compromise the security, because this Edge-to-Edge traffic is already fully inspected by the Edge devices both when the traffic flows through the Hub and when it doesn't.

Policy & Objects													
Policy Package													
ADOM: DEMO													
admin													
Policy Packages													
Search...													
Edge													
Firewall Policy													
Installation Targets													
Hub													
Firewall Policy													
Installation Targets													
default													
Firewall Policy													
Installation Targets													
Object Configurations													
Create New													
Delete													
Section													
Policy Lookup													
Collapse All													
Column Settings													
#	Name	From	To	Source	Destination	Schedule	Service	Action	NAT	Security Profiles	Log	Comment	
1	Edge-Edge	overlay hub2hub_overlay	overlay hub2hub_overlay	CORP_LAN	CORP_LAN	always	ALL	Accept	Disabled	no-inspection	Log All Sessions		
2	Edge-Hub	lan_zone hub2hub_overlay	lan_zone hub2hub_overlay	CORP_LAN	CORP_LAN	always	ALL	Accept	Disabled	default certificate-inspect	Log All Sessions		
3	Internet (DIA)	lan_zone	underlay	all	all	always	ALL	Accept	Enabled	default certificate-inspect	Log All Sessions		
4	Internet (RIA)	overlay	underlay	all	all	always	ALL	Accept	Enabled	default certificate-inspect	Log All Sessions		
5	Health-Check	overlay	Lo-HC	all	all	always	PING	Accept	Disabled	no-inspection	Log Security Events		
6	Peering	overlay hub2hub_overlay	Lo	all	all	always	PING BGP	Accept	Disabled	no-inspection	Log Security Events		
Implicit (7-7 / Total: 1)													
7	Implicit Deny	any	any	all	all	always	ALL	Deny			No Log		

## Notes about the Hub policy package

- Just like on the Edge policy package, we are using System Zones and SD-WAN Zones to keep the policy package generic. There is one additional System Zones here named *hub2hub\_overlay* for the Hub-to-Hub overlays that interconnect different regions. Our Jinja CLI Template (*Hub-MultiRegion*) will configure it on the Hub devices.
- This Policy Package is ready to support Remote Internet Access, which is traffic arriving from the Edge devices through the overlays and directed to the Internet (underlay).
- This Firewall Policy also allows Direct Internet Access for the workloads hosted behind the Hub itself.
- We must explicitly allow health-check probes that the Edge devices will send to the Hub devices, as it is done in the *Health-Check* rule.
- We must also explicitly allow incoming BGP sessions from the Edges and from the Hubs serving remote regions. (In the [BGP on loopback on page 14](#) design method, all these BGP sessions will be terminated on the main loopback interface named *Lo*). This is done in the *Peering* rule.



In the [BGP per overlay on page 13](#) design method, only the inter-regional (Hub-to-Hub) BGP peering is terminated on the loopback interface. Hence, only the *hub2hub\_overlay* zone is required in this rule.

# Deploying sites

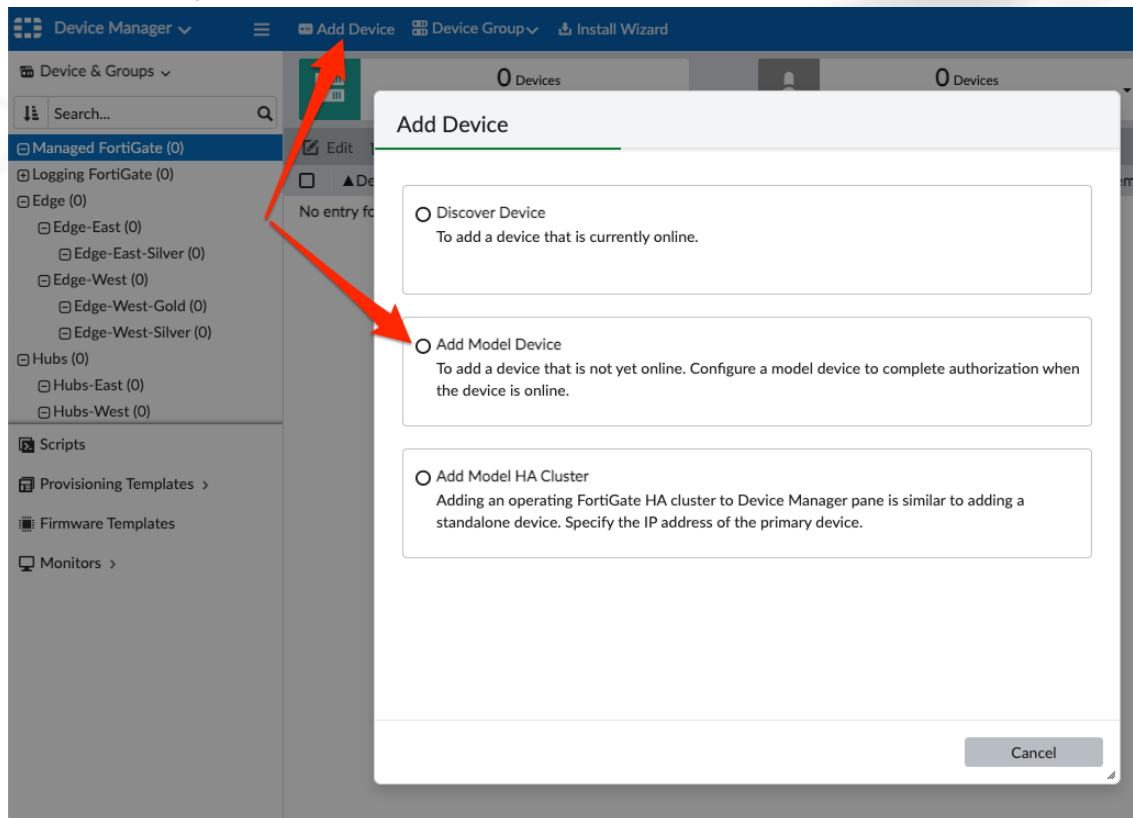
Following is a summary of how to deploy the sites:

1. Create a model device. See [Creating a model device on page 52](#).
2. Set meta field values. See [Setting meta field values on page 53](#).
3. Generate a certificate. See [Generating a certificate on page 54](#).
4. Install the configuration. See [Installing the configuration on page 55](#).
5. Install the policy packages. See [Installing policy packages on page 56](#).
6. Onboard devices. See [Onboarding devices on page 57](#).

## Creating a model device

To create a model device:

1. In *Device Manager*, click *Add Device*, and select *Add Model Device*:



2. Fill in the following details:

Field	Description
Name	Device name (e.g. hostname).
Link Device By	Chosen onboarding method. Fill in either Serial Number or Pre-shared Key.
Device Model	Device model.
Add to Device Group	Select the right group.
Pre-run CLI templates	Select the right Pre-run CLI Template.

## SETTING META FIELD VALUES

### Add Device

#### Add Model Device

Name

Link Device By ☐ Serial Number ☒ Pre-shared Key

Pre-shared Key

Device Model

☐ Enforce Firmware Version 7.0 (by default)

☒ Add to Device Group  1 Entry Selected

☐ Add to Folder

☒ Pre-Run CLI Templates

☐ Assign Policy Package

Provisioning Templates

3. Click *Next* to complete model device creation.

The new model device is associated with a list of provisioning templates. Most of the provisioning templates are automatically assigned to the model device based on the device group. An exception is the pre-run CLI template that we manually assigned when we created the model device.

Device Name	Config Status	Policy Package Status	Provisioning Templates
site1-1	Unknown	Never installed	<ul style="list-style-type: none"><li>FGTVM-initial</li><li>Basic-Settings</li><li>Edge-2H-Silver</li><li>Underlay-Static</li><li>Edge-Template</li></ul>

## Setting meta field values

Set the values of the per-device meta fields for the routing design method you chose. See [Routing design methods on page 13](#).



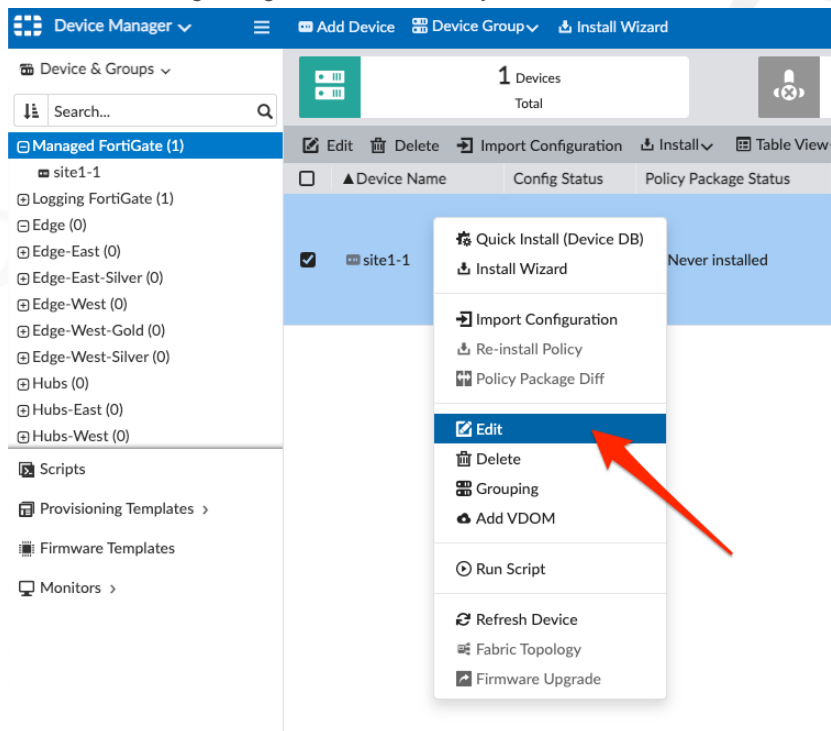
Both our design methods ([BGP per overlay on page 13](#) and [BGP on loopback on page 14](#)) currently require you to complete the same per-device meta fields. Conveniently, the list is provided in the comments in the beginning of our Project template.

For a complete reference of the Project template, please refer to the GitHub repository. See [Appendix B - External resources on page 66](#).

## GENERATING A CERTIFICATE

To set meta field values:

1. In *Device Manager*, right-click the newly added model device, and select *Edit*.



2. Fill in the values of the per-device meta fields.

lan_ip	10.0.1.1/24
loopback	10.200.1.1
mpls_wan_gateway	172.16.0.2
mpls_wan_ip	172.16.0.1/29
outbandwidth	
profile	Silver
region	West
shaping_profile	

3. Fill in the *region* and the *profile* of the device, using one of the values defined in the *Project template* file.
4. Set the location of this device on the map, and optionally set other required parameters.
5. Save the changes.

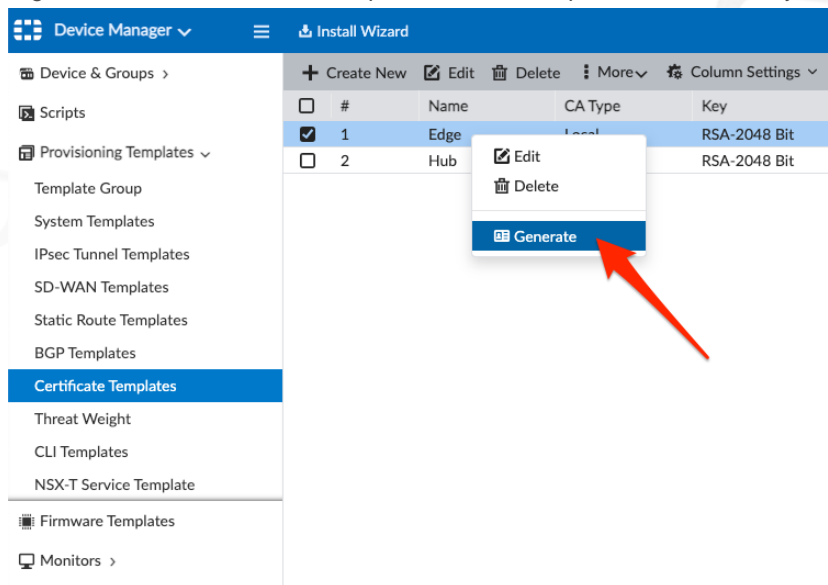
## Generating a certificate

Use the certificate template to generate a certificate and assign it to the model device.

## INSTALLING THE CONFIGURATION

To generate a certificate:

1. In *Device Manager*, go to *Provisioning Templates > Certificate Templates*.
2. Right-click the certificate template that corresponds to the newly added device, and select *Generate*:



3. Select the newly added model device, and issue the certificate for it. This certificate will be used for IKE authentication, when building IPsec overlays.

### Generate Certificates

Name: Edge

Device

<input type="checkbox"/>	Name	IP	Platform
<input checked="" type="checkbox"/>	site1-1	0.0.0.0	FortiGate-VM64-KVM

OK Cancel

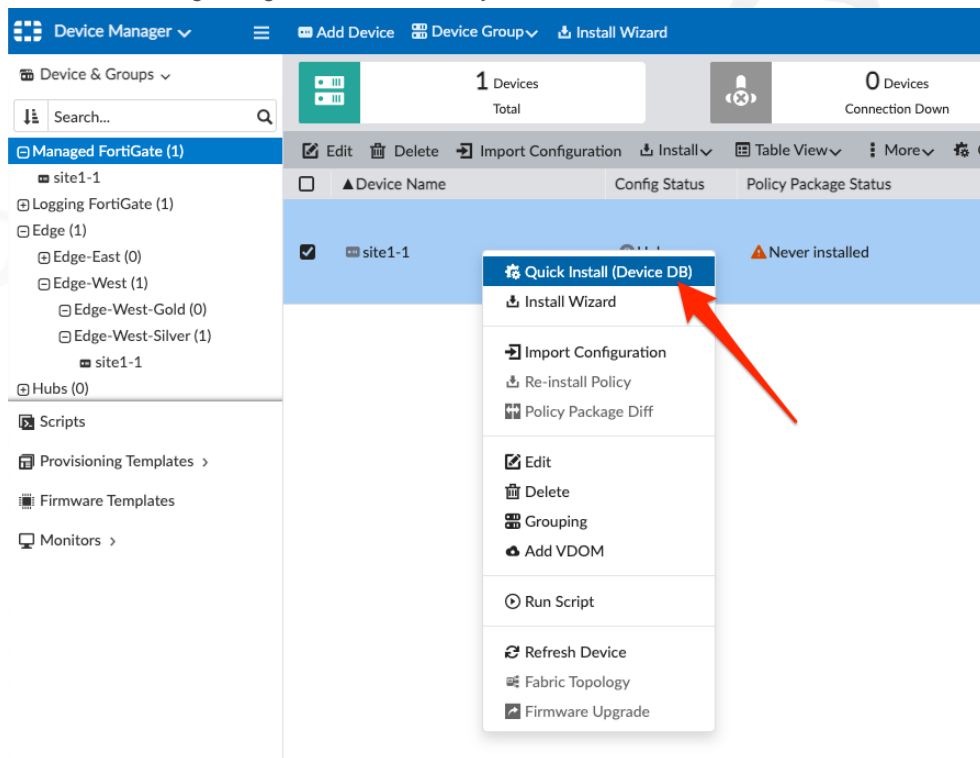
## Installing the configuration

Use the Quick Install (Device DB) method to generate the device configuration, and apply all the provisioning templates to the model device.

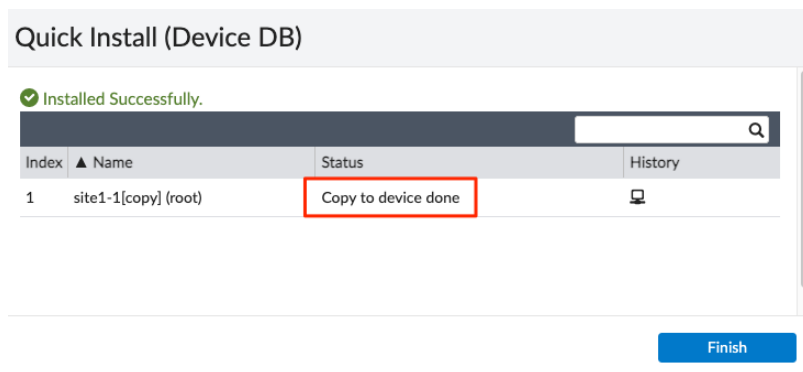
## INSTALLING POLICY PACKAGES

To install the configuration:

1. In *Device Manager*, right-click the newly added model device, and select *Quick Install (Device DB)*:



The necessary device configuration is generated, and all the assigned provisioning templates are applied.



## Installing policy packages

To install policy packages:

1. In *Device Manager*, right-click the newly added model device, and select *Install Wizard*.
2. Select *Install Policy Package & Device Settings*.
3. In the *Policy Package* list, select the firewall policy package that corresponds to this device (either *Edge* or *Hub*), and proceed with the installation.



## Install Wizard

☒ Install Policy Package & Device Settings

Install a selected policy package. Any device specific settings for devices associated with the package will also be installed.

Policy Package

Edge

Comment

0/127

☐ Create ADOM Revision☐ Schedule Install☐ Install Device Settings (only)

The model device is now ready.

Device Name	Config Status	Policy Package Status	Provisioning Templates
site1-1	Unknown	Edge	<input checked="" type="checkbox"/> Basic-Settings <input checked="" type="checkbox"/> Edge-2H-Silver <input checked="" type="checkbox"/> Underlay-Static <input checked="" type="checkbox"/> Edge-Template

## Onboarding devices

The last step is to onboard the "real" device by linking it to the respective model device. The exact actions depend on the chosen onboarding method.

- For Zero-Touch Provisioning, it is enough to connect the FortiGate device to the network and power it on. In this case, the device will be authorized using its serial number.
- For Low-Touch Provisioning, it is required to specify FortiManager details manually by using the following CLI snippet. (Device authorization can be done either using its serial number or a pre-shared key):

```
config system central-management
  set type fortimanager
  set fmg <fmg-ip>
end
```

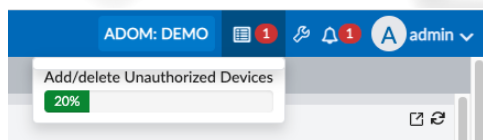
```
execute central-mgmt register-device <fmg-serial-number> <pre-shared-key>
```



Detailed overview of the FortiGate provisioning methods is outside the scope of this document.

Once the FortiGate device contacts the FortiManager, the *Auto-Link* process starts, authorizing the device and linking it to its respective model device. The complete device configuration and the firewall policy package are pushed to the device.

## ONBOARDING DEVICES



You can follow this process under *System Settings > Task Monitor*:

System Settings								
<a href="#">Group Error Devices</a> <a href="#">Delete</a> <a href="#">View Details</a> <a href="#">Show Status</a> <a href="#">Column Settings</a>								
ID	Source	Description	User	Status	Time Used	ADOM		
9	Install Configuration	Push config to device.	admin	10%	35s	DEMO		
8	Device Manager	Add/delete Unauthorized Devices	Auto link	60%	55s	DEMO		
7	Install Package	Install package to device from commit	admin	Success: 1	4s	DEMO		
6	Install Package	Copy Package 'Edge'	admin	Success: 2	6s	DEMO		
5	Install Device	Install Device	admin	Success: 1	3s	DEMO		
4	Certificate Enrollment	Signing Certificate	admin	Success: 1	4s	DEMO		
3	Device Manager	Add Device	admin	Success: 1	3s	DEMO		
2	Device Manager	Add/delete Unauthorized Devices	admin	Success: 1	<1s	root		
1	Device Manager	dvmdb adom DEMO object member	admin	Success: 1	2s	DEMO		

Once the process is complete, the FortiGate device is fully deployed and operational:

Device Manager									
<a href="#">Add Device</a> <a href="#">Device Group</a> <a href="#">Install Wizard</a>									
<a href="#">Device &amp; Groups</a> <a href="#">1 Devices Total</a> <a href="#">0 Devices Connection Down</a> <a href="#">0 Devices Device Config Modified</a>									
<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Import Configuration</a> <a href="#">Install</a> <a href="#">Table View</a> <a href="#">More</a> <a href="#">Column Settings</a>									
Device Name	Config Status	Policy Package Status	Provisioning Templates	IP Address	Platform	loopback	profile	region	
site1-1	Synchronized	Edge	<a href="#">Basic-Settings</a> <a href="#">Edge-2H-Silver</a> <a href="#">Underlay-Static</a> <a href="#">Edge-Template</a>	192.168.0.31	FortiGate-VM64-KVM	10.200.1.1	Silver	West	

---

# Additional topics

In this document, we have made several recommendations for SD-WAN configuration with regards to its interaction with the traditional routing subsystem:

- We have recommended to enable the `tie-break fib-best-match` option on all SD-WAN Rules controlling ADVPN traffic (see [Edge SD-WAN Templates]).
- We have recommended configuring all SD-WAN Zones (both "underlay" and "overlay") to act as a default route (see [Static Route Templates])

To help you better understand the logic behind these recommendations, we must explain the interaction between the **Routing** and the **SD-WAN** pillars in more detail.

Let us recap the two main rules that apply by default:

1. SD-WAN Rules are matched only if the best route to the destination points to SD-WAN.
2. SD-WAN Member is selected only if it has a valid route to the destination (not necessarily the best route).

Both these rules can be controlled using advanced options in the SD-WAN rules:

- Rule number one is controlled by the advanced option `default` (corresponding to CLI `set default disable*|enable`)
- Rule number two is controlled by the advanced options `gateway` and `tie-break` (corresponding to CLI `set gateway disable*|enable` and `set tie-break cfg-order*|fib-best-match` respectively).

Let us now look into several use cases in more detail.

## SD-WAN routing logic

In this guide, we have made several recommendations for SD-WAN configuration with regards to its interaction with the traditional routing subsystem:

- We have recommended to enable the `tie-break fib-best-match` option on all SD-WAN Rules controlling ADVPN traffic. See [Creating Edge SD-WAN templates on page 33](#).
- We have recommended configuring all SD-WAN Zones (both "underlay" and "overlay") to act as a default route. See [Static route templates on page 41](#).

To help you better understand the logic behind these recommendations, we must explain the interaction between the Routing and the SD-WAN pillars in more detail. Let us recap the two main rules that apply by default:

1. SD-WAN Rules are matched only if the best route to the destination points to SD-WAN.
2. SD-WAN Member is selected only if it has a valid route to the destination (not necessarily the best route).

Both these rules can be disabled by using advanced options in SD-WAN rules:

- Rule #1 is controlled by the advanced option `default` (corresponding to CLI `set default*|enable`).
- Rule #2 is controlled by the advanced option `gateway` (corresponding to CLI `set gateway disable*|enable` and `set tie-break cfg-order*|fib-best-match` respectively).

Let us now look into several use cases in more detail:

- [Best route mode for SD-WAN on page 60](#)
- [SD-WAN as default route on page 61](#)
- [Excluding Traffic from SD-WAN on page 63](#)

## Best route mode for SD-WAN

The rule #2 guarantees that, by default, the SD-WAN rules will not select a member that does not have a valid route to the destination. However, as we have explicitly highlighted, by default, SD-WAN rules look for **any valid route** via the member in question, even if that route is not the best route that exists in the routing table.

To be even more precise, the SD-WAN rules look for *the best available route via the member in question* (even if it is not *the best available route globally*).

For example, imagine that the SD-WAN receives a session destined to 10.4.1.1. The matched SD-WAN rule must select between the members (in the order of preference): H1\_MPLS, H2\_MPLS, H3\_MPLS.

Consider the following routing table snippet:

```
S* 0.0.0.0/0 [1/0] via 192.2.0.2, port1, [1/1]
    [1/0] via H1_MPLS tunnel 172.16.1.5, [10/1]
    [1/0] via H2_MPLS tunnel 172.16.2.5, [10/1]
    [1/0] via H3_MPLS tunnel 172.16.3.5, [10/1]
```

```
B 10.4.1.0/24 [200/0] via 10.200.1.2 (recursive via H3_MPLS tunnel 172.16.3.5)
```

Clearly, the best route to our destination is the route towards 10.4.1.0/24 via H3\_MPLS. But will the SD-WAN select this member?

According to the rule #2, the answer is: "Not by default"!

Indeed, the SD-WAN rule will check the members in the order of preference, and for each member it will look for *the best route to the destination via that member*.

What is the best route towards 10.4.1.1 via H1\_MPLS? Does it exist? Yes, it's the default route!

It is a perfectly valid route, therefore the H1\_MPLS overlay will be selected (providing that it does not violate the configured SLA target)!

If this is not the desired behavior, we can instruct the SD-WAN rule to consider only the routes that are *the best routes globally*, by configuring the advanced option `tie-break fib-best-match`:

Edit SD-WAN Rule

sla-compare-method

order

x

▼

src-negate

☐

standalone-action

☐

status

☒

tie-break

fib-best-match

x

▼

use-shortcut-sla

☒

Adding this option to the rule in the above example, will guarantee the selection of H3\_MPLS.

## Recommended for ADVPN

This advanced option is often desirable for the SD-WAN rules controlling the internal (corporate) traffic, because multiple partially overlapping summary routes are often advertised from different locations.

But above all, it is recommended for the rules controlling the Edge-to-Edge ADVPN traffic. Let's see why.

Consider a simple SD-WAN rule selecting between the two overlays (in the order of preference): H1\_ISP1, H1\_MPLS:

- A new session arrives, destined to 10.0.2.101, which belongs to the LAN prefix 10.0.2.0/24 behind a remote Edge device, advertised using BGP and thus reachable via both of the above overlays
- When the H1\_ISP1 overlay is selected and a new ADVPN shortcut H1\_ISP1\_0 is built over it, this new shortcut is automatically inserted at the head of the preference list: [ H1\_ISP1\_0, H1\_ISP1, H1\_MPLS ]
- If, however, the health of the ISP1 link on the remote Edge is degraded, this will result in this shortcut becoming out of SLA (detected by the ADVPN Shortcut Monitoring feature), and it will move to the end of the list: [ H1\_ISP1, H1\_MPLS, H1\_ISP1\_0 ]
- The desired behavior at this point is to *skip* H1\_ISP1 and select an alternative overlay H1\_MPLS (possibly building a new shortcut over it). Indeed, using the overlay H1\_ISP1 to send the traffic to the Hub would not make much sense, since the remote Edge has an unhealthy ISP1 connection!
- The ADVPN integration with BGP ensures that the best route towards 10.0.2.0/24 via the shortcut will *hide* the corresponding route via the parent tunnel H1\_ISP1. And therefore, there will be *no best route* to the destination via H1\_ISP1.
- However, similar to the previous example, there may still be a default route (or another summary route) via H1\_ISP1, which by default would be considered valid enough for the SD-WAN rule to select this member!
- To avoid this, we recommend configuring `tie-break fib-best-match`, to guarantee that only *the best* route towards 10.0.2.0/24 is considered.

## SD-WAN as default route

According to the rule #2, by default, SD-WAN rules select a member only if there is a valid route to the destination through that member. For Edge-to-Hub and Edge-to-Edge traffic, this valid route will normally be learned by way of BGP. However, for Edge-to-Internet traffic there will be no specific route learned. Hence, for example, in order for the **Remote Internet Access** to work as desired in our examples, it is required to have a default gateway via the MPLS overlays (H1\_MPLS, H2\_MPLS). Otherwise the traffic destined to the Internet would never be backhauled to the Hubs.

Configuring SD-WAN to act as a default route for the "overlay" zone solves this problem. Furthermore, it eliminates the need to adjust the routing configuration whenever your SD-WAN rules change. Simply put, it ensures that there always be a valid route to any destination via any SD-WAN member that is selected by the SD-WAN rules. Thus, SD-WAN rules become fully responsible for the traffic steering, in accordance with the

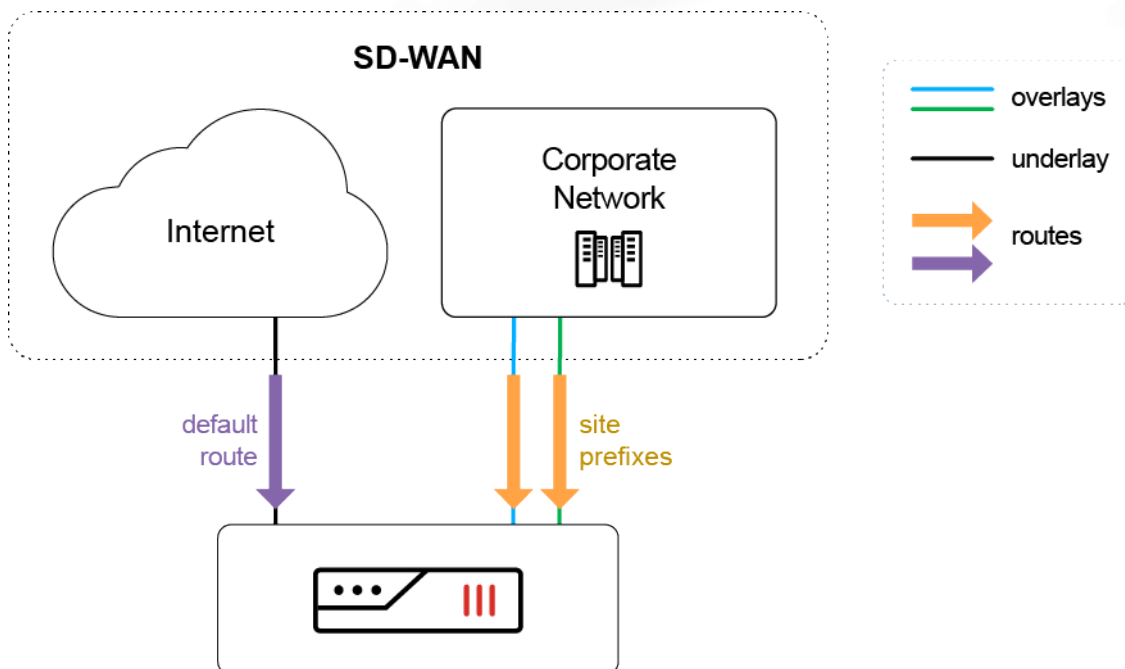
**Five-Pillar Design Approach.** For these reasons, we have recommended this configuration throughout this document.

Nevertheless, it is worth noting a few alternatives to this approach:

- [Using the default route via underlay on page 62](#)
- [Disabling route check in SD-WAN rules on page 62](#)

## Using the default route via underlay

If RIA functionality is not required, it is enough to only keep the default route by way of the "underlay" zone only:



- Edge-to-Internet traffic will be able to breakout locally, using this default route.
- Edge-to-Hub and Edge-to-Edge traffic will use the specific routes learned through BGP.

No other routing configuration is needed.



If any other destination exists that must be reached through the overlays (either Internet destination or a legacy site outside the SD-WAN solution), you must ensure that a valid route to that destination exists through the required overlay. This route can be either learned dynamically or configured statically.

## Disabling route check in SD-WAN rules

Another alternative to using SD-WAN as a default route globally is to disable route check on per-rule basis.

For each SD-WAN rule where a valid route to the destination is not expected to exist (such as the RIA rules), you can enable the two advanced options `gateway` and `default`, as mentioned in the note in [Using the default route via underlay on page 62](#).

## Create New SD-WAN Rule

### Advanced Options ▾

addr-mode	ipv4 ▾
bandwidth-weight	0
default	<input checked="" type="checkbox"/> ON
dscp-forward	<input type="checkbox"/> OFF
dscp-forward-tag	000000
dscp-reverse	<input type="checkbox"/> OFF
dscp-reverse-tag	000000
dst-negate	<input type="checkbox"/> OFF
gateway	<input checked="" type="checkbox"/> ON
hash-mode	round-robin ▾
hold-down-time	0
input-device	Click here to select
input-device-negate	<input type="checkbox"/> OFF
jitter-weight	0
latency-weight	0
link-cost-threshold	10

OK

Cancel

This instructs the SD-WAN rule to bypass any route check, and forward the traffic unconditionally through the member selected by the configured strategy. Hence, if TH1\_MPLS is selected in our RIA example, the Internet traffic will be backhauled to the Hub, even if there is no default route learned through H1\_MPLS.

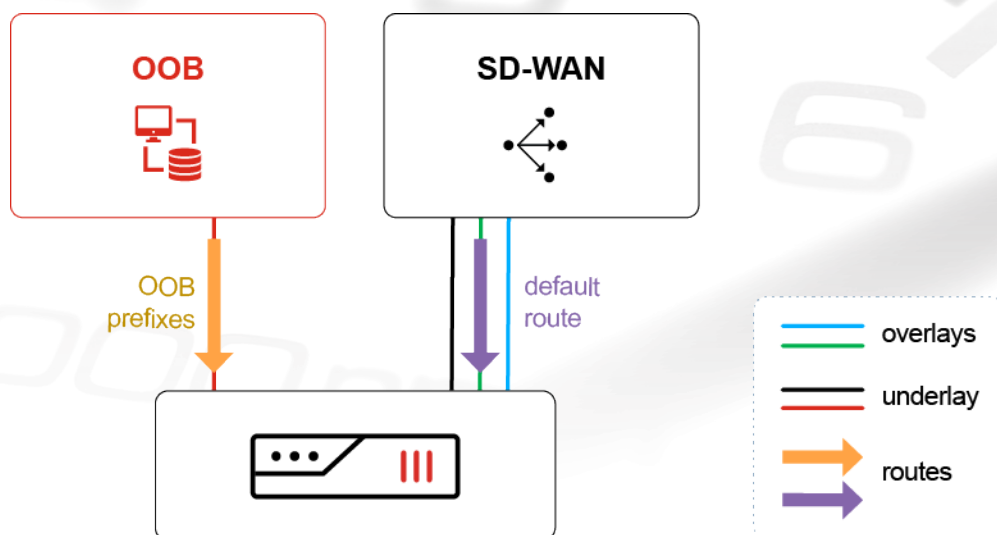
## Excluding Traffic from SD-WAN

Note that even if you configure SD-WAN to act as a default route, as recommended throughout this guide, you can still exclude certain traffic from SD-WAN processing. The rule #1 helps you to achieve that.

By default, before selecting a member, SD-WAN rule will check whether the best route to the destination points to *any* SD-WAN member at all. If it doesn't, this traffic will be considered as *out of SD-WAN scope*, and hence it will be handled by the traditional routing.

A good use case is out-of-band management (OOB). As shown on the below diagram, SD-WAN acts as a default route, but there are more specific management prefixes learned via the OOB network.

## SD-WAN ROUTING LOGIC



This guarantees that all the traffic destined to those prefixes bypasses SD-WAN rule processing and is forwarded to the OOB network.



# Appendix A - Products used

The following product models and firmware were used in this guide:

Product	Model	Firmware
FortiOS	All models supported by FortiManager	7.0.6
FortiManager	All models	7.0.4

# Appendix B - External resources

Following are the GitHub repositories mentioned throughout this document:

- Reference set of Jinja Templates (<https://github.com/fortinet-solutions-cse/sdwan-advpn-reference/tree/release/7.0>)
- Automation collection for Postman ([https://github.com/fortinet-solutions-cse/postman\\_collections/tree/7.0.x](https://github.com/fortinet-solutions-cse/postman_collections/tree/7.0.x))



[www.fortinet.com](http://www.fortinet.com)



Copyright© 2024 Fortinet, Inc. All rights reserved. Fortinet®, FortiGate®, FortiCare® and FortiGuard®, and certain other marks are registered trademarks of Fortinet, Inc., and other Fortinet names herein may also be registered and/or common law trademarks of Fortinet. All other product or company names may be trademarks of their respective owners. Performance and other metrics contained herein were attained in internal lab tests under ideal conditions, and actual performance and other results may vary. Network variables, different network environments and other conditions may affect performance results. Nothing herein represents any binding commitment by Fortinet, and Fortinet disclaims all warranties, whether express or implied, except to the extent Fortinet enters a binding written contract, signed by Fortinet's Chief Legal Officer, with a purchaser that expressly warrants that the identified product will perform according to certain expressly-identified performance metrics and, in such event, only the specific performance metrics expressly identified in such binding written contract shall be binding on Fortinet. For absolute clarity, any such warranty will be limited to performance in the same ideal conditions as in Fortinet's internal lab tests. Fortinet disclaims in full any covenants, representations, and guarantees pursuant hereto, whether express or implied. Fortinet reserves the right to change, modify, transfer, or otherwise revise this publication without notice, and the most current version of the publication shall be applicable.