



# Pulse Secure Virtual Traffic Manager: Configuration Importer Guide

Supporting Pulse Secure Virtual Traffic Manager 19.2

Product Release	<b>19.2</b>
Published	<b>15 July, 2019</b>
Document Version	<b>1.0</b>

Pulse Secure, LLC  
2700 Zanker Road,  
Suite 200 San Jose  
CA 95134

[www.pulsesecure.net](http://www.pulsesecure.net)

© 2019 by Pulse Secure, LLC. All rights reserved.

Pulse Secure and the Pulse Secure logo are trademarks of Pulse Secure, LLC in the United States. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Pulse Secure, LLC assumes no responsibility for any inaccuracies in this document. Pulse Secure, LLC reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

*Pulse Secure Virtual Traffic Manager: Configuration Importer Guide*

The information in this document is current as of the date on the title page.

## **END USER LICENSE AGREEMENT**

The Pulse Secure product that is the subject of this technical documentation consists of (or is intended for use with) Pulse Secure software. Use of such software is subject to the terms and conditions of the End User License Agreement (“EULA”) posted at <http://www.pulsesecure.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

# Contents

---

PREFACE .....	1
DOCUMENT CONVENTIONS .....	1
TEXT FORMATTING CONVENTIONS .....	1
COMMAND SYNTAX CONVENTIONS .....	1
NOTES AND WARNINGS .....	2
REQUESTING TECHNICAL SUPPORT .....	2
SELF-HELP ONLINE TOOLS AND RESOURCES .....	2
OPENING A CASE WITH PSGSC .....	3
INTRODUCTION .....	5
ABOUT THE CONFIGURATION IMPORTER .....	5
ONE-TIME IMPORT .....	6
FULL CONFIGURATION MANAGEMENT .....	6
BASIC USAGE .....	7
INTRODUCING CONFIGURATION DOCUMENTS .....	7
STRUCTURING CONFIGURATION DOCUMENTS FOR IMPORT .....	8
ONE-TIME CONFIGURATION IMPORT EXAMPLE .....	9
EXAMPLE DOCKER DEPLOYMENT .....	9
EXAMPLE FOR STANDARD DEPLOYMENTS .....	10
FULLY MANAGED CONFIGURATION EXAMPLE .....	10
EXAMPLE DOCKER DEPLOYMENT .....	10
EXAMPLE WITH STANDARD DEPLOYMENTS .....	11
OBJECT REFERENCES .....	11
TROUBLESHOOTING .....	13
SYNTAX ERRORS .....	13
SEMANTIC ERRORS .....	14
AUDIT LOGS .....	14
RELATIONSHIP TO OTHER CONFIGURATION INTERFACES .....	17
COMPARISON WITH THE REST API .....	19
YAML vs JSON .....	19
OBJECT STRUCTURE .....	19
CREATING MULTIPLE CONFIGURATION OBJECTS .....	20
DEFINING CONFIGURATION SPECIFIC TO A TRAFFIC MANAGER INSTANCE .....	20

UNSTRUCTURED RESOURCES .....	21
VALIDATION.....	21
IMPORTING CONFIGURATION INTO A CLUSTER .....	23
REPLICATING CONFIGURATION .....	23
INDEPENDENT CLUSTER MEMBERS .....	23
CHANGING SETTINGS THAT REQUIRE A RESTART.....	25
SOFTWARE RESTARTS .....	25
HOST INSTANCE REBOOTS.....	25
CONSTRUCTING CONFIGURATION DOCUMENTS.....	27
CONSTRUCTING CONFIGURATION DOCUMENTS MANUALLY .....	27
EXPORTING A CONFIGURATION DOCUMENT FROM THE ADMIN UI .....	27
EXPORTING A CONFIGURATION DOCUMENT FROM THE COMMAND-LINE.....	29
LAYERING CONFIGURATION DOCUMENTS.....	31
OBJECT REFERENCES .....	35
REFERENCING EXTERNAL OBJECTS .....	35
SUPPORTED VALUEFROM METHODS.....	36
CHANGES TO REFERENCED OBJECTS.....	36
EXAMPLE: IMPORTING TLS CERTIFICATES FROM KUBERNETES SECRETS .....	36
SNAPSHOTS.....	39
INTRODUCING CONFIGURATION SNAPSHOTS.....	39
SNAPSHOTTING CONFIGURATION IN DOCKER.....	39
SNAPSHOTTING CONFIGURATION IN OTHER DEPLOYMENTS .....	41
STATEFUL SETTINGS.....	43
UPGRADES.....	45
IMPORTING CONFIGURATION TO UPGRADED TRAFFIC MANAGERS.....	45
UPGRADING TRAFFIC MANAGERS WITH IMPORTED CONFIGURATION.....	45
CONFIGURATION DOCUMENT VERSIONING .....	47

# Preface

---

- [Document conventions](#) ..... 1
- [Requesting Technical Support](#) ..... 2

## Document conventions

The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in Pulse Secure technical documentation.

### Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used in the flow of the text to highlight specific words or phrases.

Format	Description
<b>bold text</b>	Identifies command names
	Identifies keywords and operands
	Identifies the names of user-manipulated GUI elements
	Identifies text to enter at the GUI
<i>italic text</i>	Identifies emphasis
	Identifies variables
	Identifies document titles
Courier Font	Identifies command output
	Identifies command syntax examples

### Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
<b>bold text</b>	Identifies command names, keywords, and command options.
<i>italic text</i>	Identifies a variable.
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.

Convention	Description
{ <b>x</b>   <b>y</b>   <b>z</b> }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
<b>x</b>   <b>y</b>	A vertical bar separates mutually exclusive elements.
< >	Non-printing characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, member[member...].
\	Indicates a “soft” line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

## Notes and Warnings

Note, Attention, and Caution statements might be used in this document.

**Note:** A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

### ATTENTION

An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

### CAUTION

A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

## Requesting Technical Support

Technical product support is available through the Pulse Secure Global Support Center (PSGSC). If you have a support contract, file a ticket with PSGSC.

- Product warranties—For product warranty information, visit <https://support.pulsesecure.net/product-service-policies/>

## Self-Help Online Tools and Resources

For quick and easy problem resolution, Pulse Secure provides an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <https://support.pulsesecure.net>
- Search for known bugs: <https://support.pulsesecure.net>
- Find product documentation: <https://www.pulsesecure.net/techpubs>
- Download the latest versions of software and review release notes: <https://support.pulsesecure.net>
- Open a case online in the CSC Case Management tool: <https://support.pulsesecure.net>

- To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://support.pulsesecure.net>

For important product notices, technical articles, and to ask advice:

- Search the Pulse Secure Knowledge Center for technical bulletins and security advisories: <https://kb.pulsesecure.net>
- Ask questions and find solutions at the Pulse Community online forum: <https://community.pulsesecure.net>

## Opening a Case with PSGSC

You can open a case with PSGSC on the Web or by telephone.

- • Use the Case Management tool in the PSGSC at <https://support.pulsesecure.net>.
- • Call 1-844 751 7629 (Toll Free, US).

For international or direct-dial options in countries without toll-free numbers, see <https://support.pulsesecure.net/support/support-contacts/>





# Introduction

This chapter introduces the Pulse Secure Virtual Traffic Manager (the Traffic Manager) *Configuration Importer* tool, and describes the deployment scenarios within which it is intended be used.

This chapter contains the following sections:

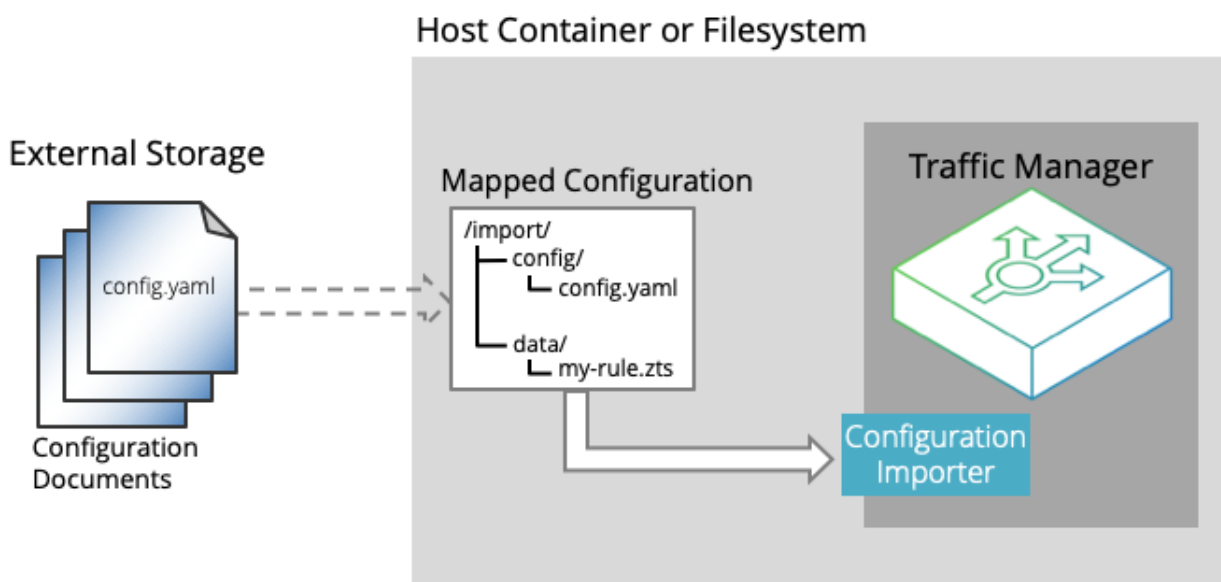
- [About the Configuration Importer](#) ..... 5
- [One-Time Import](#) ..... 6
- [Full Configuration Management](#) ..... 6

## About the Configuration Importer

The Configuration Importer is a tool that accepts administrator-supplied *configuration documents* and imports the configuration described in those documents into the Traffic Manager, replacing its previous running configuration. Configuration documents are intended to define the entire Traffic Manager configuration.

Configuration documents are typically stored in a system external to the Traffic Manager, and mounted or mapped into the Traffic Manager host's file system such that the Configuration Importer can read them.

Figure 1 Mapping configuration documents to the Traffic Manager



The Configuration Importer supports two primary modes of operation:

- A one-time import
- Full configuration management

To learn more about both modes of operation, read the remainder of this chapter.

## One-Time Import

The Configuration Importer tool can initialize or reset a Traffic Manager's configuration to a pre-defined state in a single operation. This is known as a one-time import.

An administrator or orchestration tool (such as Docker or Kubernetes) might use a one-time import to deploy and configure a Traffic Manager in a single step. For example, a Traffic Manager Docker container can be deployed in a fully configured state from a single "docker run" command.

The one-time import can also be used to reset a running Traffic Manager's configuration to a specific state.

## Full Configuration Management

The Traffic Manager can use the Configuration Importer tool for fully-managed configuration updates, where the Traffic Manager monitors mapped configuration documents and automatically updates its configuration when a change is detected.

In this mode of operation, the configuration documents become the primary definition of the entire Traffic Manager configuration. The Configuration Importer detects whenever the primary configuration documents change and automatically updates the Traffic Manager configuration accordingly.

Instead of making iterative sequential changes through the Traffic Manager's Admin UI or APIs, an administrator makes changes by editing the configuration documents to the desired new state.

This externally-maintained declarative configuration is often appropriate for environments that use automated deployment and orchestration tools. If a Traffic Manager needs to be destroyed and re-created, the information needed to recreate its configuration is not lost because the primary copy is held in an external location.

Configuration documents can be stored in, and deployed from, a version control system supporting a DevOps style of working that reduces the need for direct interaction with individual Traffic Manager instances.

# Basic Usage

This chapter provides an overview of how to use the Configuration Importer. It contains the following sections:

- [Introducing Configuration Documents](#) ..... 7
- [Structuring Configuration Documents for Import](#) ..... 8
- [One-time Configuration Import Example](#) ..... 9
- [Fully Managed Configuration Example](#) ..... 10
- [Object References](#) ..... 11

## Introducing Configuration Documents

Configuration documents describe Traffic Manager configuration in a syntax that allows an administrator to create, read, alter, and maintain configuration definitions within version control systems. The configuration is expressed in either YAML (YAML Ain't Markup Language) or JSON (JavaScript Object Notation) format, based upon the REST API schema defined in the *Pulse Secure Virtual Traffic Manager: REST API Guide*.

The following example configuration document describes a simple Traffic Manager configuration, containing a single virtual server and a TrafficScript rule, in both YAML and JSON formats:

### example-config.yaml

```
version: 6.1

virtual_servers:
- name: example-service
  properties:
    basic:
      enabled: true
      port: 80
      protocol: http
      pool: discard
      request_rules: [ basic-response ]

rules:
- name: basic-response
  content: |
    http.sendResponse("200 OK", "text/plain",
      "Hello World\n",
      "X-Served-By: Pulse Secure vTM");
```

### example-config.json

```
{
  "version": 6.1,
  "virtual_servers": [
    {
      "name": "example-service",
      "properties": {
        "basic": {
          "enabled": true,
          "port": 80,
          "protocol": "http",
          "pool": "discard",
          "request_rules": [
            "basic-response"
          ]
        }
      }
    }
  ],
  "rules": [
    {
      "name": "basic-response",
      "content": "http.sendResponse(\"200 OK\",
        \"text/plain\", \"Hello World\\n\",
        \"X-Served-By: Pulse Secure vTM\");\n"
    }
  ]
}
```

Configuration documents have a few key differences to the REST API format. The following table lists some key concepts and a comparison of how they are implemented in configuration documents and the REST API:

Concept	Configuration Documents	REST API
Configuring Multiple Objects	A configuration document can describe many resources of different types together in a single document.	Each object has its own API endpoint. Multiple API calls are needed to configure multiple objects.
Unstructured Resources	The content for unstructured resources, such as TrafficScript rules, is expressed inline in the "content" field.	The resource is uploaded to the object's API endpoint with a content-type of "octet-stream" instead of JSON.
Declarative VS Imperative	Configuration documents express the desired state of the Traffic Manager in its entirety.	REST API calls make incremental changes to the existing configuration.

For a more detailed description of the differences between configuration documents and the REST API, see ["Comparison with the REST API" on page 19](#).

**Note:** To learn more about YAML, see [www.yaml.org](http://www.yaml.org). To learn more about JSON, see [www.json.org](http://www.json.org).

## Structuring Configuration Documents for Import

The Configuration Importer runs from a base directory and looks for configuration documents in a subdirectory named "config". Other supporting files and directories can be added to the base directory and referenced from the configuration documents. For more details on referencing external files from within configuration documents, see ["Object References" on page 35](#).

The following is an example directory structure:

```

/import/
├── config/
│   ├── configuration-document-1.yaml
│   ├── configuration-document-2.yaml
│   └── ...
├── data/
│   ├── my-rule.zts
│   └── catalog.json
├── secrets/
│   └── db-password
├── tls/
│   └── example-service/
│       ├── tls.key
│       └── tls.crt

```

When the Configuration Importer is fully managing the configuration, changes to any files under the base directory result in the Traffic Manager's configuration being updated.

The base directory must be visible in the file system of the Traffic Manager. The Configuration Importer expects that the contents of the directory are populated from an external system; for example, being mounted or mapped into the Traffic Manager's filesystem from a persistent volume, or pulled from an external repository.

## One-time Configuration Import Example

The following example shows how to perform a one-time configuration import within Docker, and additionally how to invoke the Configuration Importer on standard deployments.

Save the following sample YAML configuration document as the file `./import/config/example-config.yaml`:

### example-config.yaml

```
version: 6.1

virtual_servers:
- name: example-service
  properties:
    basic:
      enabled: true
      port: 80
      protocol: http
      pool: discard
      request_rules: [ basic-response ]

rules:
- name: basic-response
  content: |
    http.sendResponse("200 OK", "text/plain", "Hello World\n", "X-Served-By: Pulse Secure vTM");
```

## Example Docker Deployment

When deploying the Traffic Manager container, mount the "import" directory as a volume and specify the mount path in the ZEUS\_BASE\_CONFIG environment variable. After installing the Traffic Manager, the container's startup script searches for configuration documents within the "config" subdirectory of the mounted volume and performs a one-time import of the configuration documents contained there.

To deploy a fully configured Traffic Manager inside a Docker container, use the following command:

```
# Assumes the sample configuration document is saved as
# /import/config/example-config.yaml
```

```
docker run --name=vtm-config-example \
  -v `pwd`/import:/import \
  -p 8080:80 \
  -p 9090:9090 \
  -e ZEUS_BASE_CONFIG=/import \
  -e ZEUS_EULA=accept \
  -e ZEUS_PASS=admin \
  --privileged \
  --init \
  -t \
  -d \
  pulsesecure/vtm:19.2
```

The above example launches a Traffic Manager container named "vtm-config-example" that is populated with the configuration specified in "/import/config/example-config.yaml". To verify that the deployment was successful, make an HTTP request to port 8080 on your Traffic Manager's primary IP address or hostname. This should result in a "Hello World" response.

To see that the configuration has been correctly applied, access the Administration Interface (Admin UI) of the Traffic Manager instance on port 9090.

## Example for Standard Deployments

Standard software or virtual appliance deployments of the Traffic Manager include the Configuration Importer tool as standard, but without built-in support for invoking it at install time.

An administrator can invoke the Configuration Importer manually from the command-line using the following command:

```
# Assumes the sample configuration document is saved as
# /import/config/example-config.yaml

$ZEUSHOME/zxtm/bin/config-import --chdir /import ./config
```

In this example, the Configuration Importer uses "/import" as its base directory and imports the configuration specified in any documents found in the "/import/config" directory.

## Fully Managed Configuration Example

The following example shows how to use the Configuration Importer to fully manage a Traffic Manager's configuration within Docker, and additionally how to invoke the Configuration Importer on standard deployments.

The example given here uses the same sample YAML configuration document shown in the one-time import example, saved at the same filesystem location: "./import/config/example-config.yaml".

## Example Docker Deployment

When you deploy a Traffic Manager container, mount the import directory as a volume and specify the mount path in the ZEUS\_WATCHED\_CONFIG environment variable. After installing the Traffic Manager, the container imports the configuration documents stored in the "config" subdirectory of the mounted volume. The Configuration Importer continuously watches for changes in the ZEUS\_WATCHED\_CONFIG directory and automatically imports the updated configuration when such a change is detected.

To deploy a fully configuration-managed Traffic Manager inside a Docker container, use the following command:

```
# Assumes the sample configuration document is saved as
# /import/config/example-config.yaml

docker run --name=vtm-config-example \
```

```

-v `pwd`/import:/import \
-p 8080:80 \
-p 9090:9090 \
-e ZEUS_WATCHED_CONFIG=/import \
-e ZEUS_EULA=accept \
-e ZEUS_PASS=admin \
--privileged \
--init \
-t \
-d \
pulsesecure/vtm:19.2

```

The above example launches a Traffic Manager container named "vtm-config-example" that is populated with the configuration specified in the watched directory (in this case, "/import/config/example-config.yaml"). To verify that the deployment was successful, make an HTTP request to port 8080 on your Traffic Manager's primary IP address or hostname. This should result in a "Hello World" response.

To see that the configuration has been correctly applied, access the Administration Interface (Admin UI) of the Traffic Manager instance on port 9090.

Next, edit "/import/config/example-config.yaml" and change the rules section to:

```

rules:
- name: basic-response
  content: |
    http.sendResponse("200 OK", "text/plain", "Hello New World\n", "X-Served-By: Pulse
Secure vTM");

```

Observe that the configuration has been updated by making a further HTTP request to port 8080, which should now return "Hello New World".

## Example with Standard Deployments

To watch for changes to configuration documents with software or virtual appliance deployments, use the "watch-directory" tool supplied with the Traffic Manager. Specify the directory to watch, followed by the command to run when any files in the directory change.

To watch for changes in the "/import" directory, and then apply those changes, run the following command:

```

$ZEUSHOME/zxtm/bin/watch-directory /import -- $ZEUSHOME/zxtm/bin/config-import \
--chdir /import ./config

```

## Object References

To dynamically fetch and include configuration values from external files into your configuration documents, use the "valueFrom" construct. Use this mechanism to store items such as TrafficScript rules and SSL certificates outside the main configuration document in their native format.

For example, a TrafficScript rule can be extracted into a separate file and accessed through "valueFrom", as follows:

#### `/import/data/hello-world.zts`

```
http.sendResponse("200 OK", "text/plain", "Hello New World\n", "X-Served-By: Pulse Secure vTM");
```

---

#### `/import/config/example-config.yaml`

```
version: 6.1

virtual_servers:
- name: example-service
  properties:
    basic:
      enabled: true
      port: 80
      protocol: http
      pool: discard
      request_rules: [ basic-response ]

rules:
- name: basic-response
  content:
    valueFrom:
      fileRef:
        name: data/hello-world.zts
```

---

For more details on how to reference external data from configuration documents, see [“Object References” on page 35](#).



# Troubleshooting

This chapter contains the following sections:

- [Syntax Errors](#) ..... 13
- [Semantic Errors](#) ..... 14
- [Audit Logs](#) ..... 14

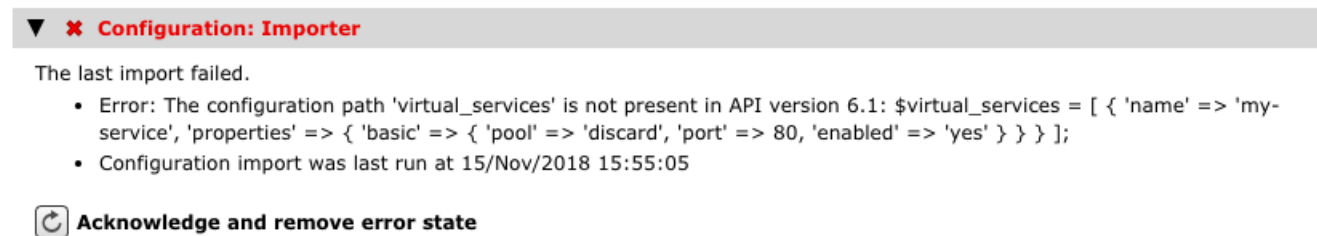
## Syntax Errors

The Configuration Importer tool checks the syntax of configuration documents as they are processed. If there are syntax errors in any configuration document, the Configuration Importer detects them and reports an error in the container logs without changing the Traffic Manager's configuration.

The Traffic Manager also displays this error on the Diagnose page of the Admin UI.

To dismiss a reported import error and return the Traffic Manager to a normal running state, click **Acknowledge and remove error state**. An error is cleared automatically if a subsequent successful import occurs.

Figure 2 Configuration Importer error output on the Diagnose page.



To view import error details when applying configuration to a Docker container, view the container logs.

For example, for a container named "vtm-config-example", run the command:

```
docker logs vtm-config-example
```

Figure 3 Example error output from a Docker container log

```

Removing inactive config 'conf_B'
Creating new config 'conf_B' from snapshot
Error: The configuration path 'virtual_services' is not present
in API version 6.1:
$virtual_services = [
  {
    'name' => 'my-service',
    'properties' => {
      'basic' => {
        'pool' => 'discard',
        'port' => 80,
        'enabled' => 'yes'
      }
    }
  }
];

```

In this example, the configuration document incorrectly refers to "virtual\_services" instead of "virtual\_servers".

## Semantic Errors

If a configuration document contains semantic errors such as a required setting not being present, the Configuration Importer continues to import the configuration to the Traffic Manager with an error shown in the import log. The Traffic Manager also shows a configuration error in the event log and Diagnose page.

To check if the configuration was applied as you expected, see the Configuration Importer log after you trigger an import:

Figure 4 Semantic errors reported in the Configuration Importer log (or container error log for Docker deployments)

```

Removing inactive config 'conf_B'
Creating new config 'conf_B' from snapshot
Creating /usr/local/zeus/zxtm/conf_B/vservers/foo
Set 'conf_B' as the active configuration
There are configuration errors:
SERIOUS vservers/foo port
Required config key 'port' is missing
Tags: confrequired

```

### ATTENTION

Pulse Secure recommends frequently checking the Traffic Manager's event log and Diagnose page to ensure your configuration has been deployed correctly.

## Audit Logs

Successful and unsuccessful import attempts are logged to the Traffic Manager's Audit Log.

Figure 5 The Audit Log

	Timestamp	User	Group	Auth	IP	Description
✓	15/Nov/2018:16:01:45 +0000	root	root	console	-	Config import completed successfully
✓	15/Nov/2018:16:01:45 +0000	root	root	console	-	File added: <b>vservers/my-service</b>
✓	15/Nov/2018:16:01:44 +0000	root	root	console	-	User initiated a config import: <b>config</b>



# Relationship to Other Configuration Interfaces

---

The Traffic Manager has a number of other configuration interfaces, such as the Admin UI, the REST API, the CLI, and the SOAP API. All of these mechanisms rely on the Traffic Manager being deployed and then iteratively configured until it is in the desired state.

When fully managing the configuration of your Traffic Manager, the Configuration Importer is an alternative to, and a replacement for, these other configuration interfaces. The administrator makes changes by altering the primary configuration documents, automatically triggering the updated configuration to be applied to the Traffic Manager. The other configuration interfaces still provide the current state of the configuration and the health of the Traffic Manager, but an administrator should not normally update the configuration through these interfaces as any such changes are overwritten when the next import is triggered.

During a one-time import, an administrator uses the Configuration Importer in conjunction with the other configuration interfaces. After the one-time import supplies the initial configuration, the administrator makes any further iterative changes through the Traffic Manager's usual configuration interfaces.

Importing configuration is a similar operation to restoring a backup, however the format of the imported configuration is easier to maintain and version control than a backup file and follows the well-documented REST API configuration schema.

The Configuration Importer supports layering of configuration documents to allow multiple sources of Traffic Manager configuration to be merged together in a declarative way. This capability is similar to the Traffic Manager's "Partial Backups" feature and the "zconf" utility, both of which support backup and restore of subsets of the Traffic Manager configuration. However, partial backups do still build configuration iteratively and have an internal format that is not as suitable for editing and maintenance as configuration documents.

For more details on how to layer configuration across multiple documents, see ["Layering Configuration Documents" on page 31](#).



# Comparison with the REST API

---

Configuration documents follow the same configuration schema as the Traffic Manager's REST API, with a few notable differences described here.

This chapter contains the following sections:

- [YAML vs JSON](#) ..... 19
- [Object Structure](#) ..... 19
- [Creating Multiple Configuration Objects](#) ..... 20
- [Defining Configuration Specific to a Traffic Manager Instance](#) ..... 20
- [Unstructured Resources](#) ..... 21
- [Validation](#) ..... 21

## YAML vs JSON

To write your configuration documents, use either YAML or JSON format. The Traffic Manager REST API supports only JSON. The examples used in this guide are based around YAML due to its human-readable format and suitability for storing configuration in a version control system.

## Object Structure

When creating a new object using the REST API, you define the object's properties in the body data of an HTTP PUT request. The URL defines the API version and name of the new object.

For example, to create a new IP-based session persistence class using the REST API, you use a PUT request with the following object to an endpoint such as `/api/tm/7.0/config/active/persistence/my_persistence_class`:

```
{
  "properties": {
    "basic": {
      "type": "ip"
    }
  }
}
```

The equivalent configuration document in YAML is as follows:

```
version: 6.1
persistence:
- name: my_persistence_class
  properties:
    basic:
      type: ip
```

The configuration definition in both cases contains a top level object named "properties", with all the configuration keys nested below it in one or more sections.

All settings defined within the "properties" section follow the REST schema. To view the schema, see the *Pulse Secure Virtual Traffic Manager: REST API Guide*.

## Creating Multiple Configuration Objects

The REST API requires a separate API call for each object you want to create, whereas a single configuration document can hold the definition for the entire Traffic Manager configuration.

Configuration documents can contain sections for multiple object types, and multiple objects of the same type can be specified as list items beneath the object type definition (using the "-" array item identifier in YAML, or array syntax in JSON). For example:

```
version: 6.1

bandwidth:
- name: my_bandwidth_class
  properties:
    basic:
      maximum: 10000
      sharing: machine

persistence:
- name: my_ip_persistence_class
  properties:
    basic:
      type: ip
- name: my_transparent_persistence_class
  properties:
    basic:
      type: transparent
```

## Defining Configuration Specific to a Traffic Manager Instance

Configuration documents can include configuration specific to individual Traffic Managers through the "traffic\_managers" section. Each named object in this section refers to the hostname of a Traffic Manager instance in the cluster. Such configuration can include, for example, networking or other system level settings.

To identify the local Traffic Manager on which you run the import, without naming it explicitly, use the special object name "local\_tm". This feature is useful for orchestration platforms such as Docker or Kubernetes where you might not know the instance hostname at the point you construct the configuration document. For example:

```
version: 6.1

traffic_managers:
- name: local_tm
```



```
properties:
  snmp:
    community: body
    enabled: true
    security_level: noauthnopriv
```

## Unstructured Resources

Some configuration objects are not expressed as key/value pairs, but rather inline as raw text. Examples of these include TrafficScript rules, SSL certificates, and custom monitor scripts.

When using the REST API, you upload such configuration objects using a PUT request with a content-type of "application/octet-stream".

To present the raw data in a configuration document, use a "content" field. Declare the data either inline or as an indented block using a YAML literal scalar (designated by the '|' indicator). The following example expresses a TrafficScript rule:

```
rules:
- name: basic-response
  content: |
    http.sendResponse("200 OK", "text/plain", "Hello World\n", "X-Served-By: Pulse Secure vTM");
```

For binary files, content can be base64 encoded with an "encoding: base64" field added to the object definition. For details of the supported base64 type requirements, see RFC2045 (<https://www.ietf.org/rfc/rfc2045.txt>).

Alternatively, you can import values from an external file using the object reference syntax described in [“Object References” on page 11](#).

## Validation

The Configuration Importer always applies a configuration that is syntactically correct. A syntactically correct configuration with a semantic error, such as a port number being out of the valid port range, is still applied, but the Traffic Manager reports an error and does not start the affected service.

This behavior is designed to ensure that the Traffic Manager's configuration is consistent with the expected outcome of invoking the Configuration Importer, regardless of any historical state. The Configuration Importer does not assume that the entity that generated the configuration is checking to make sure it was accepted by the Traffic Manager.

In contrast, the REST API applies validation of individual configuration settings and rejects a PUT request if a setting is invalid. The REST API assumes that whatever invoked the PUT request is also checking to make sure it completed successfully.



# Importing Configuration into a Cluster

---

When running as a cluster, all Traffic Managers must have an identical copy of the configuration. When using the Configuration Importer, the Traffic Manager supports two alternative methods of ensuring the configuration of all cluster members remains consistent:

- [Replicating Configuration](#) ..... 23
- [Independent Cluster Members](#) ..... 23

## Replicating Configuration

When an administrator makes a configuration change through the Admin UI or the REST API, they select a specific cluster member on which to make the change. The Traffic Manager then automatically replicates that change out to all other members of the cluster.

The Configuration Importer can also operate under the same principle. An administrator selects a cluster member on which to run the Configuration Importer, and the imported configuration is automatically replicated out to all other cluster members. When configuration is entirely managed through the Configuration Importer, you designate a single cluster member to watch for configuration changes and apply them to the whole cluster.

## Independent Cluster Members

As an alternative, provided each Traffic Manager has access to the same configuration documents, you can instruct the Configuration Importer to not replicate configuration to the other cluster members. In this situation, each cluster member must maintain its own configuration independently.

For Docker deployments, to have each container independently manage its own configuration, launch the container with the `ZEUS_CONFIG_IMPORT_ARGS` environment variable set to `--no-replicate`.

For other deployments, add the `--no-replicate` argument to the `config-import` tool when it is invoked. For example:

```
$ZEUSHOME/zxtm/bin/watch-directory /import -- $ZEUSHOME/zxtm/bin/config-import \
--chdir /import --no-replicate ./config
```



# Changing Settings that Require a Restart

---

This chapter discusses the difference between Traffic Manager software restarts and full host reboots. It contains the following sections:

- [Software Restarts](#) ..... 25
- [Host Instance Reboots](#) ..... 25

## Software Restarts

The Configuration Importer automatically restarts the Traffic Manager software on all cluster members if a changed configuration setting requires a restart to take effect.

In some circumstances, you might want to disable this behavior and delay restarts to happen at a later time, such as during a maintenance window. To disable automatic restarting of the software, supply the "--no-restart" argument to the "config-import" tool, or add "--no-restart" to the ZEUS\_CONFIG\_IMPORT\_ARGS environment variable when launching a Docker container. By using this argument, you must manually restart your Traffic Managers at the appropriate time for pending changes to take effect.

## Host Instance Reboots

Some configuration changes might require a full host instance reboot to take effect. This is not an automatic process and must be performed manually on all cluster members. Pulse Secure recommends you schedule a full reboot to take place at a time of least impact to your services.



# Constructing Configuration Documents

---

This chapter describes how to set up configuration documents suitable for use with the Configuration Importer. It contains the following sections:

- [Constructing Configuration Documents Manually](#) ..... 27
- [Exporting a Configuration Document From the Admin UI](#) ..... 27
- [Exporting a Configuration Document From the Command-line](#) ..... 29

## Constructing Configuration Documents Manually

To construct a configuration document suitable for use with the Configuration Importer, Pulse Secure recommends using the following approaches:

- **Using the REST API Guide:** Consult the Pulse Secure Virtual Traffic Manager: REST API Guide to find all the available configuration objects and their properties. This information can be used to construct appropriate configuration documents to represent the desired Traffic Manager configuration.
- **Admin UI:** Build and test the configuration using the Admin UI, then perform REST API calls to retrieve the configuration for each object and write it into a configuration document.

## Exporting a Configuration Document From the Admin UI

To obtain a pre-constructed configuration document representing the configuration of the current Traffic Manager, click **Services > Configuration Summary** in the Admin UI. To generate a configuration document, use the "Export Configuration document" section.

**Note:** This functionality is also available for previously-saved configuration backups, through the **System > Backups** page.

Figure 6 Exporting a configuration document representing the current configuration

### Export Configuration document

Export your current configuration as a configuration document. Configuration documents are suitable for version control and offline review. A configuration document can be imported to a new traffic manager using the configuration import tool.

The file format to export.

**Format:**  **YAML** A human readable text format, for easy review and editing.

**JSON** A machine readable text format, widely supported by scripting languages.

Whether to export the configuration changes made since initial configuration (differences), or the full configuration.

**Export Type:**  **Differences** Configuration which has changed since initial configuration.

**Full** The full configuration, including built in configuration that has never been edited.

How to deal with secrets such as passwords and TLS private keys. Unless secrets are included, the configuration document cannot be directly imported without editing.

**Secrets:**  **Include** ! Secrets are included, and the export should be kept secure.

**Exclude** Secrets are replaced with a null value.

**Hash** ! Secrets are replaced with a hash of their content, the export should still be kept secure.

To modify the appearance of a configuration document, use the following options:

Option	Value	Description
Format	YAML	Create the configuration document in human-readable YAML text format.
	JSON	Create the configuration document in machine-readable JSON text format.
Export Type	Differences	Include only those configuration changes made <i>since</i> the Traffic Manager was first configured.
	Full	Include all configuration present on the Traffic Manager, irrespective of whether the configuration was set during initial configuration or at some time since.



Option	Value	Description
Secrets	Include	<p>Include all secret data, such as passwords and TLS private keys.</p> <p>A configuration document created with this option is complete and ready to be used with the Configuration Importer, but is unsafe to share publicly.</p> <p><b>Note:</b> Pulse Secure strongly recommends taking additional security measures when creating configuration documents with this option selected.</p>
	Exclude	<p>Redact all secret data.</p> <p>A configuration document created with this option is safe to share publicly (as allowed by your organizational security policy), or to check into a version control system. However, to properly configure a new Traffic Manager instance from this configuration document, you must first edit in all required secret data items.</p>
	Hash	<p>Replace all secret data items with a hash of their contents.</p> <p>Use this option to compare configuration document versions and to observe when secret data might have changed, without revealing the secrets to casual observers.</p> <p><b>Note:</b> Pulse Secure recommends that a document created with this option should be kept securely to minimize the risk of offline password cracking and other such attacks.</p>

To generate a configuration document based on the selected options, click one of the following buttons:

- To download the configuration document to your local workstation, click **Download**.
- To display the configuration document in your browser, click **View in Browser**.

## Exporting a Configuration Document From the Command-line

An administrator can also generate a configuration document from the command-line using the Traffic Manager's `config-export` program. For example:

```
# Exports a configuration document to /import/config/example-config.yaml
$ZEUSHOME/zxtm/bin/config-export --filename "example-config.yaml"
                                --format yaml
                                --secrets exclude
                                --chdir /import
                                ./config
```

For a complete list of the available options, type:

```
$ZEUSHOME/zxtm/bin/config-export --help
```



# Layering Configuration Documents

---

The Configuration Importer can collect together and import multiple configuration documents at the same time if instructed to import from a directory. The Configuration Importer orders the documents alphabetically, and then constructs the configuration by layering the documents one on top of the other. You can build up configuration for the same object, such as a virtual server, over multiple documents, with later documents masking settings applied by earlier documents.

The following two example configuration documents reference the same "example-service" virtual server object. The "10-base-service.yaml" file is applied first and creates the base configuration for the service. The file "20-set-rules.yaml" is alphabetically second in order of priority and layered on top of the base configuration to add a TrafficScript rule to the virtual server, and to disable web caching.

## 10-base-service.yaml

```
version: 6.1

virtual_servers:
- name: example-service
  properties:
    basic:
      enabled: true
      port: 80
      protocol: http
      pool: example-pool
    web_cache:
      enabled: true

pools:
- name: example-pool
  properties:
    basic:
      passive_monitoring: true
      monitors: [ "Simple HTTP" ]
      nodes_table:
        - node: application-server-1:8080
        - node: application-server-2:8080
```

---

## 20-set-rules.yaml

```

version: 6.1

virtual_servers:
- name: example-service
  properties:
    basic:
      request_rules: [ set-host ]
    web_cache:
      enabled: false

rules:
- name: set-host
  content: |
    http.setHeader("Host", "app1.example.com");

```

---

Importing the two documents applies configuration equivalent to the following single document:

## flattened-config.yaml

```

version: 6.1

virtual_servers:
- name: example-service
  properties:
    basic:
      enabled: true
      port: 80
      protocol: http
      pool: example-pool
      request_rules: [ set-host ]
    web_cache:
      enabled: false

pools:
- name: example-pool
  properties:
    basic:
      passive_monitoring: true
      monitors: [ "Simple HTTP" ]
      nodes_table:
        - node: application-server-1:8080
        - node: application-server-2:8080

rules:
- name: set-host
  content: |
    http.setHeader("Host", "app1.example.com");

```

---

Observe that both files specify the "virtual\_servers.properties.web\_cache.enabled" setting, although the Configuration Importer applies the "false" value in "20-set-rules.yaml" last, so this is the resultant setting applied to the Traffic Manager.

**Note:** Pulse Secure recommends limiting overlaying of specific settings within an object to basic scalar values. Layering semantics for non-scalar settings, such as lists or tables, is presently unspecified.



# Object References

This chapter describes how to reference external data and objects from within your configuration definition. It contains the following sections:

- [Referencing External Objects](#) ..... 35
- [Supported valueFrom Methods](#) ..... 36
- [Changes to Referenced Objects](#) ..... 36
- [Example: Importing TLS Certificates from Kubernetes Secrets](#) ..... 36

## Referencing External Objects

The Configuration Importer enables you to reference externally-stored data in your configuration documents. This method can help keep your configuration documents neater, and allows configuration such as TrafficScript rules and certificates to be stored in their native format. It also allows data to be pulled in from other sources when needed. For example, in a Kubernetes environment, passwords and certificates could be mounted from Secret objects and referenced inside your Traffic Manager configuration definition.

To reference externally-stored data in your configuration, use the "valueFrom" construct. The example that follows shows how to import a TrafficScript rule stored in a separate file:

### `/import/config/rule-config.yaml`

```
rules:
- name: example-rule
  content:
    valueFrom:
      fileRef:
        name: data/example-rule.zts
```

### `/import/data/example-rule.zts`

```
log.info("Hello World");
http.sendResponse("200 OK", "text/plain", "Hello World", "X-Served-By: Pulse Secure vTM");
```

The Configuration Importer replaces the valueFrom statement in the configuration document with the literal text of the rule and applies the resulting configuration to the Traffic Manager.

## Supported valueFrom Methods

The Configuration Importer supports the following valueFrom methods:

### Supported valueFrom Methods

fileRef	Import all data from the file found at the path specified in the "name" property. Path references are relative to the directory from which the Configuration Importer was invoked, or relative to the "--chdir" argument if one was specified. When launching a Docker container, path references are relative to the ZEUS_BASE_CONFIG or ZEUS_WATCHED_CONFIG environment variable.
---------	--

## Changes to Referenced Objects

If a referenced object is inside the directory being watched for changes, a subsequent change to the object causes the Configuration Importer to automatically update the Traffic Manager's configuration. If the referenced object is outside the watched directory, changes are picked up the next time the Configuration Importer runs for any other reason.

## Example: Importing TLS Certificates from Kubernetes Secrets

A common way to manage TLS keys and certificates in Kubernetes is to store them in *Secret* objects and then mount those secrets into the container that uses them. In Kubernetes, mounting a TLS Secret results in the private key being mounted as "tls.key" and the certificate being mounted as "tls.crt". For example, the configuration document and TLS files for the service could be mounted as follows:

```
/import/
├── config/
│   └── vtm-config.yaml
└── tls/
    └── example-service/
        ├── tls.key
        └── tls.crt
```



Based on this file structure, the following configuration document configures a virtual server to present the certificate and decrypt the incoming traffic:

### vtm-config.yaml

```
virtual_servers:
- name: example-service
  properties:
    basic:
      enabled: true
      port: 443
      protocol: http
      pool: example-service-pool
      ssl_decrypt: true
    ssl:
      server_cert_default: example-service-cert

ssl:
  server_keys:
  - name: example-service-cert
    properties:
      basic:
        public:
          valueFrom:
            fileRef:
              name: tls/example-service/tls.crt
        private:
          valueFrom:
            fileRef:
              name: tls/example-service/tls.key
```

---



# Snapshots

---

This chapter discusses configuration snapshots and how to create them in Docker and other environments. It contains the following sections:

- [Introducing Configuration Snapshots](#) ..... 39
- [Snapshotting Configuration in Docker](#) ..... 39
- [Snapshotting Configuration in Other Deployments](#) ..... 41

## Introducing Configuration Snapshots

The Traffic Manager takes a snapshot of its configuration when it is first deployed. This *base configuration snapshot* incorporates all the environment-specific configuration that is established during initial configuration, such as the host's timezone, the ports on which administrative services are listening, and tunings to support particular cloud environments.

When the Configuration Importer runs, the Traffic Manager's configuration is reset to this initial state before the imported configuration is applied.

If you need to change this base configuration, you can apply static configuration to your Traffic Manager such as logging endpoints, security settings, and cache sizes, and then create an updated snapshot. Your changes become part of the base configuration snapshot and do not need to be specified in imported configuration documents.

## Snapshotting Configuration in Docker

If you deploy a Docker container with the ZEUS\_BASE\_CONFIG environment variable set, the Traffic Manager creates a configuration snapshot after the base configuration has been applied. Any additional configuration imported from the directory specified in the ZEUS\_WATCHED\_CONFIG environment variable is layered on top of this base configuration.

In practice, this is relevant only if the base configuration and watched configuration directories are different, such that the watched configuration is applied on top of the base configuration. For example.:

```
/base/config/system-settings.yaml
```

```
version: 6.1

global_settings:
  properties:
    session:
      ip_cache_size: 65535
    web_cache:
      size: 2GB
      max_file_size: 50MB
```

---

## /watched/config/my-service.yaml

```

version: 6.1

virtual_servers:
- name: example-service
  properties:
    basic:
      enabled: true
      port: 80
      protocol: http
      pool: discard
      request_rules: [ basic-response ]

rules:
- name: basic-response
  content: |
    http.sendResponse("200 OK", "text/plain", "Hello World\n", "X-Served-By: Pulse Secure vTM");

```

Deploying a container in Docker using these configuration documents results in the base configuration in "system-settings.yaml" being applied to the Traffic Manager initially, the base configuration snapshot being updated, and then the configuration in "my-service.yaml" being applied on top of that base configuration. Any changes to configuration in the "watched" directory are applied on top of the base configuration. However, changes to configuration in the "base" directory after the container is launched have no effect as that directory is not being watched for changes.

To deploy the container, use the following command

## Deploy container with base and watched configuration

```
# Assumes the sample configuration documents shown above are saved as ./base/config/system-
settings.yaml and ./watched/config/my-service.yaml
```

```

docker run --name=vtm-config-example \
  -v `pwd`/base:/base \
  -v `pwd`/watched:/watched \
  -p 8080:80 \
  -p 9090:9090 \
  -e ZEUS_BASE_CONFIG=/base \
  -e ZEUS_WATCHED_CONFIG=/watched \
  -e ZEUS_EULA=accept \
  -e ZEUS_PASS=admin \
  --privileged \
  --init \
  -t \
  -d \
  pulsesecure/vtm:19.2

```

## Snapshotting Configuration in Other Deployments

For other deployment types, take a snapshot after the Traffic Manager has been deployed to create a new base configuration snapshot.

To snapshot the current Traffic Manager configuration, run the following command:

```
$ZEUSHOME/zxtm/bin/config-snapshot
```



# Stateful Settings

---

A small number of configuration settings are considered "stateful" and are not reset to a previous value when reverting to the snapshot configuration. Such settings are generally managed by an external entity, such as Pulse Secure Services Director.

You can override stateful settings in configuration documents, but if the setting is later removed from the configuration document its value is not reverted in the Traffic Manager's configuration.

Some examples of stateful settings that the Configuration Importer does not reset include:

- An FLA license pushed by the Services Director
- Analytics Export configuration
- Management-plane user groups
- Management-plane user authenticators
- The cluster ID





# Upgrades

---

This chapter discusses the effect of Traffic Manager version upgrades on the Configuration Importer. It contains the following sections:

- [Importing Configuration to Upgraded Traffic Managers](#)..... 45
- [Upgrading Traffic Managers with Imported Configuration](#)..... 45

## Importing Configuration to Upgraded Traffic Managers

As described in [“Snapshots” on page 39](#), the Configuration Importer requires a snapshot of the configuration when the Traffic Manager is first deployed in order to establish a baseline on which to apply the imported configuration. As such, the Configuration Importer is not able to apply configuration to a Traffic Manager that was upgraded from a version prior to 19.2 as no such snapshot was taken at the time the Traffic Manager was first deployed.

Pulse Secure recommends you deploy a new Traffic Manager instance at the latest version and use the Configuration Importer with the new instance. However, if necessary, create a snapshot manually on the upgraded instance using the instructions described in [“Snapshotting Configuration in Other Deployments” on page 41](#).

## Upgrading Traffic Managers with Imported Configuration

Pulse Secure recommends that upgrades are performed by creating a new Traffic Manager instance at the latest version and using the Configuration Importer to apply the same configuration as applied to the older instance.



# Configuration Document Versioning

---

Each configuration document contains a "version" field that references the API schema version for the configuration in the document. The Traffic Manager into which the configuration is imported must support the API schema version of the configuration in the document.

Where possible, the Configuration Importer translates configuration from the version of the API schema referenced in the configuration document to the latest supported API schema version. If the conversion cannot be performed (for example, if a configuration setting is removed from the API), the Configuration Importer logs a warning and does not alter the current running configuration.

**Note:** Version 19.2 of the Traffic Manager supports only API schema version 7.0.

