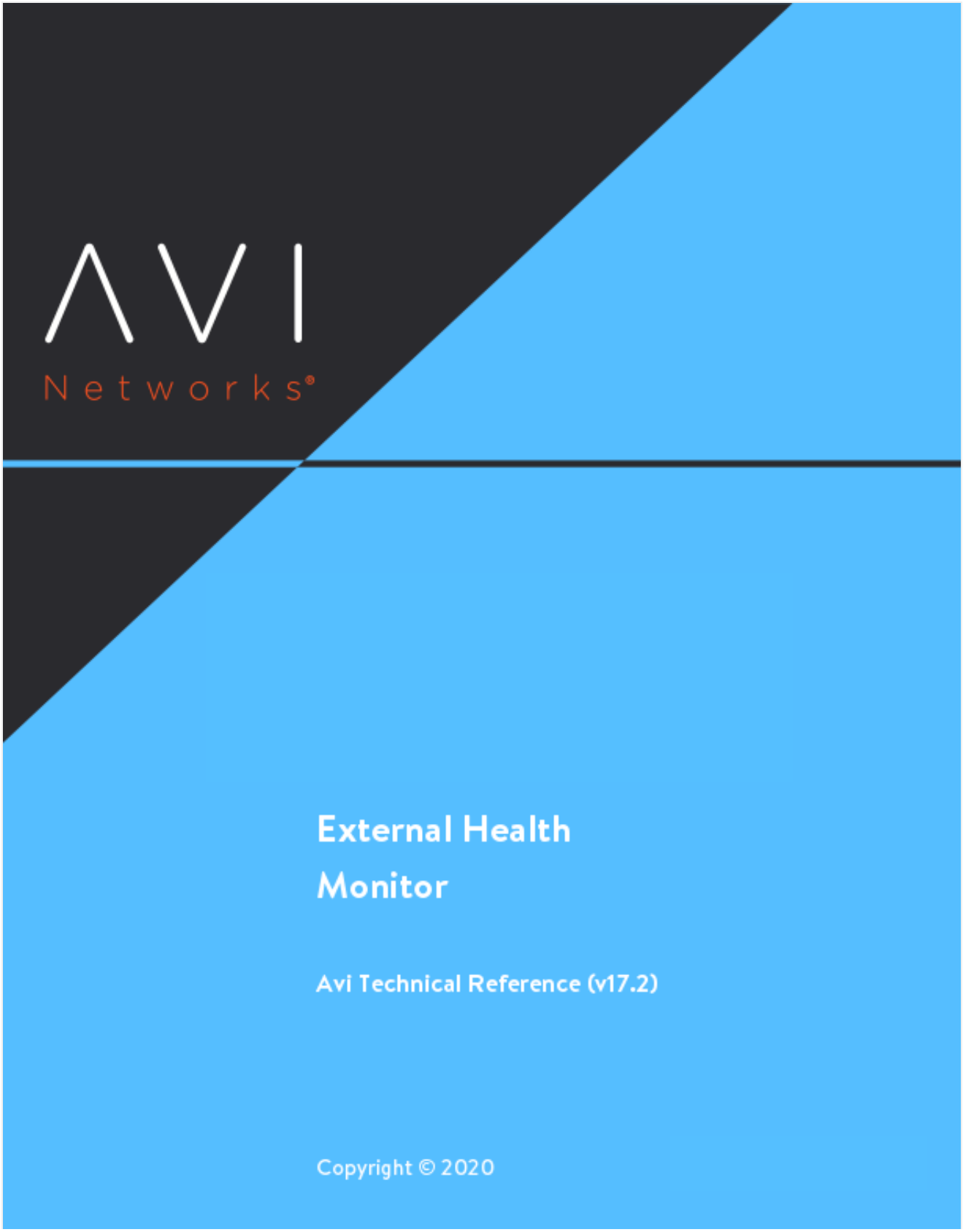

External Health Monitor

Avi Technical Reference (v17.2)

External Health Monitor

view online

This article covers the specific configuration for this monitor type. See the Overview of Health Monitors article for general monitor information, implementation, and other monitor types.

The external monitor type allows scripts to be written to provide highly customized and granular health checks. The scripts may be Linux shell, Python, or Perl, which can be used to execute `wget`, `netcat`, `curl`, `snmpget`, `mysql-client`, or `dig`. External monitors have constrained access to resources, such as CPU and memory, so as to not affect normal functioning of Avi Service Engines. As with any custom scripting, thoroughly validate the long term stability of the implemented script before pointing it at production servers.

Errors generated from the script may be viewed in the output of the Operations > Events log.

Avi Vantage includes three sample scripts via the System-Xternal Perl, Python and Shell monitors.

General Monitor Configuration

- Send Interval: Frequency at which the health monitor initiates a server check, in seconds.
  - Best practice: For busy Service Engines, keep the monitoring interval lower and receive timeout larger, since external checks tend to use more system resources than the system default monitors.
- Receive Timeout: Maximum amount of time before the server must return a valid response to the health monitor, in seconds.
- Successful Checks: Number of consecutive health checks that must succeed before Avi Vantage marks a down server as being back up.
- Failed Checks: Number of consecutive health checks that must fail before Avi Vantage marks an up server as being down.

External Specific Configuration

As a best practice, clean up any temporary files created by scripts.

When building an external monitor, it is common to manually test the successful execution of the commands. To test a command from an SE, it may be necessary to switch to the proper namespace or tenant. The production external monitor will correctly use the proper tenant. To manually switch tenants when testing a command from the SE CLI, follow the commands in the following article: Manually Validate Server Health.

- Script Code: Upload the script via copy/paste or uploading the file.
- Script Parameters: Enter any optional arguments to apply. These strings are passed in as arguments to the script, such as $1 = server IP, $2 = server port.
- Script Variables: Custom environment variables may be fed into the script to allow for simplified re-usability. For example, a script that authenticates to the server may have a variable set to USER=test.
- Script Success: If a script exits with any data, it is considered a success and marks a server up. If there is no data from the script, the monitor will mark the server down. In the SharePoint monitor example below, the script includes a `grep "200 OK"`. If this is found, this data is returned and the monitor exits as a success. If the `grep` does not find this string, no data is returned and the monitor marks the server down.

MySQL example script:

```
#!/bin/bash
#mysql --host=$IP --user=root --password=s3cret! -e "select 1"
```

SharePoint example script:

```
#!/bin/bash
#curl http://$IP:$PORT/Shared%20Documents/10m.dat -I -L --ntlm -u $USER:$PASS -I -L > /run/hmuser/$HM_NAME.out 2>/dev/r
curl http://$IP:$PORT/Shared%20Documents/10m.dat -I -L --ntlm -u $USER:$PASS -I -L | grep "200 OK"
```

SharePoint script variables:

```
USER='foo\administrator' PASS=foo123
```

Oracle example script:

```
#!/usr/bin/python
import sys
import os
import cx_Oracle
IP=os.environ['IP']
conn_str='HR_user/HR_pw@' + IP  + '/hr_db'
connection = cx_Oracle.connect(conn_str)
cursor = connection.cursor()
cursor.execute('select * from JOBS')
for row in cursor:
    print row
connection.close()
```

Oracle script variables:

```
LD_LIBRARY_PATH=/usr/lib/oracle/12.2/client64/lib
```

RADIUS example script:

The below example performs an "Access-Request" using PAP authentication against the RADIUS pool member and checks for an "Access-Accept" response.

```
#!/usr/bin/python
import os
import radius

try:
    r = radius.Radius(os.environ['RAD_SECRET'],
```

```
                    os.environ['IP'],
                    port=int(os.environ['PORT']),
                    timeout=int(os.environ['RAD_TIMEOUT']))
    if r.authenticate(os.environ['RAD_USERNAME'], os.environ['RAD_PASSWORD']):
        print 'Access Accepted'
except:
    pass
```

RAD_SECRET, RAD_TIMEOUT, RAD_USERNAME and RAD_PASSWORD can then be passed in the health monitor script variables, for example:

```
RAD_SECRET=foo123 RAD_USERNAME=avihealth RAD_PASSWORD=bar123 RAD_TIMEOUT=1
```

List of SE Packages

Scripting Languages

- Bash (shell script)
- Perl
- Python

Linux Packages (`apt`)

- curl
- snmp
- dnsutils
- libpython2.7
- python-dev
- mysql-client
- nmap
- freetds-dev
- freetds-bin

Python Packages (`pip`)

- pymssql
- cx_Oracle (and related libraries for Oracle Database 12c) ? Avi Vantage 17.1.3 onwards
- py-radius ? Avi Vantage 17.2.5 onwards