Enov8 Open Agents

January 2021

Introduction

Enov8 Open agents are agents written to automate EcoSystem Intelligence gathering.

Open Agents are atomic (focused on a particular task), send information back to EcoSystem via the REST API and are completely free to use. Agents are also “completely open source” and you can extend or utilize the code as necessary.

The current Open Agent Family includes:

Health Agents

• Component Health	Checks service on a component is active.
• Lean Synthetics	Checks URL / WEB health.
• API Health	Checks API Health.

Discovery Agents

• Component Discovery	NMAP discovery of network component (e.g. servers).
• Service Discovery	NMAP discovery of running services i.e. Open Ports.

Other “Local” Agents

• Trip Wire	Monitor for “illegal” changes to a server via filesystem changes.
• Glances Agent	“On Server” Agent to capture Insights like Process & Resource Usage

Agent Code

Currently all open agents are written in Python. However, the agents are atomic enough & simple enough to be easily read & easily converted into your scripting language of choice.

Agent Installation

Below are installation steps for installing the "Open" (python) agents.

1. Install python v3.7.x (Windows or Unix)
2. Download agent from: https://www.enov8.com/download-versions/
3. Execute the following pip command from the agent folder:

   pip install –no-index –find-links requirements -r requirements\requirements.txt
   Note: you may need to run this command instead
   python -m pip install –no-index –find-links requirements -r requirements\requirements.txt

4. Configure the config.ini with correct information (e.g. ecosystem location and API details)
5. Execute the <agent>.py script and check the log folder for the output
6. Setup a scheduler

   Use Native Scheduler, i.e. Windows Scheduler or Cron, to run agent regularly.

   Note: Scheduling although optional is recommended and ensures insights are up-to date.

   Note: We suggest waking up & executing every 30 minutes.

Agent Binaries

Enov8 also provides pre-built binaries (.exe) in the download area.

- https://www.enov8.com/download-versions/

FYI: Windows & Linux ".exe" (binary) versions can be easily created by using python tools like PyInstaller or py2exe. This has benefits of hiding code and is a great way to hide associated configuration details like your API key.

Health Agents

Component Health

Introduction

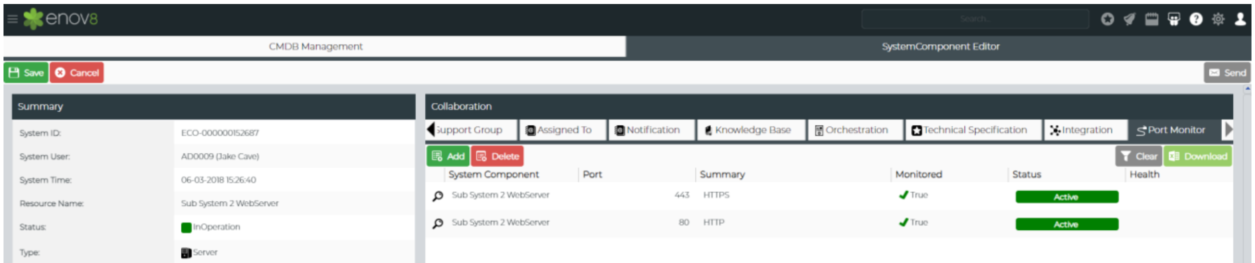
The agent checks the health of components registered inside EcoSystem using the assumption that every component has at least one critical heartbeat. For example, if the monitored component was a Web Server it would be active on port 80/443 and as a result it may have multiple heart beats. This script continually checks heartbeat is "UP" & updates Ecosystem.

How it Works

1. Agent calls the Enov8 platform & fetches active* Ecosystem components that have a "Port Monitor".

Note*: The agent ignores deleted or decommissioned components.

Note: There is Port Monitor configuration area inside the System Component Editor (detailed view).



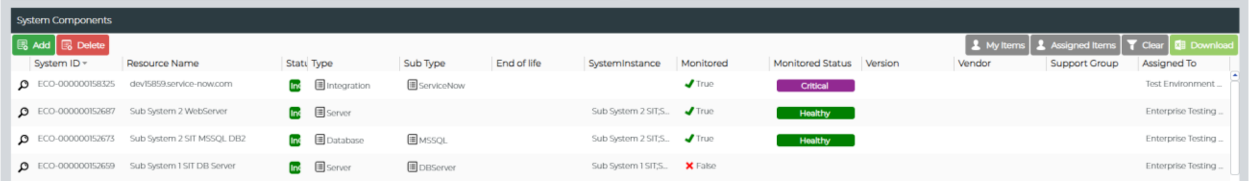
System ID:	ECO-000000152687
System User:	AD0009 (Jake Cave)
System Time:	06-03-2018 15:26:40
Resource Name:	Sub System 2 WebServer
Status:	🟩 InOperation
Type:	Server

System Component	Port	Summary	Monitored	Status	Health
Sub System 2 WebServer	443	HTTPS	✓ True	Active	
Sub System 2 WebServer	80	HTTP	✓ True	Active	

2. Agent checks that port(s) are up or down.

3. Agent updates "**Monitored Status**" accordingly.

Update EcoSystem port monitor status if it is UP or DOWN.



System ID	Resource Name	Statu	Type	Sub Type	End of life	SystemInstance	Monitored	Monitored Status	Version	Vendor	Support Group	Assigned To
ECO-000000158325	dev15859.service-now.com	In	Integration	ServiceNow			✓ True	Critical				Test Environment ...
ECO-000000152687	Sub System 2 WebServer	In	Server			Sub System 2 SIT;S...	✓ True	Healthy				Enterprise Testing ...
ECO-000000152673	Sub System 2 SIT MSSQL DB2	In	Database	MSSQL		Sub System 2 SIT;S...	✓ True	Healthy				Enterprise Testing ...
ECO-000000152659	Sub System 1 SIT DB Server	In	Server	DBServer		Sub System 1 SIT;S...	✗ False					Enterprise Testing ...

Configuration

The agent configuration can be edited via config.ini

```
[APIDetails]
;Example API details - please replace with your information
app_id=EXAMPLE_ID
app_key=EXAMPLE_KEY
user_id=EXAMPLE_USER
enov8_url=https://example/ecosystem/
```

```
default_org=ECO-000000007898
;The amount of time (in hours) that must pass for the agent to be deemed InActive
failover_dur=24

[TestOptions]
;Set ping to true to enable ping test, otherwise set it to false
ping=false
```

API Details:	
app_id	Unique API ID that can be created in ecosystem by a system admin
app_key	Unique API Key is used with the above ID for authentication
user_id	Ecosystem User ID (This user will be the author of all changes)
enov8_url	The exact URL of your ecosystem instance
default_org	The system ID of the organisation that objects will be connected to
failover_dur	The amount of time (in hours) that must pass for the agent to be deemed InActive

Test Options:	
ping	If this attribute is enabled the ping test will be performed.

Execution

Open Source Version: <my-python-path>/python.exe component_health.py

Binary Version: component_health.exe

Note: Once working you should schedule* agent to wake up & run periodically.

Lean Synthetics

Introduction

This agent is a quick way to determine, remotely, whether your web application is up & running. Most systems today have a 2/3 tier architecture, typically a Web server, an application layer and a database. And the easiest way to determine quickly if the system instance is running is to follow the following simple “Lean Shakedown steps”:

(1) Browse to Welcome Page

(2) Login (optional)

(3) Search for some data / keyword

Tip! Try to choose data / keyword that would be derived from the back-end database.

How it Works

The agent will visit a specified URL, check for a keyword & then send the results* to the Enov8 platform.

*The results are stored as a tag using the following notation:

- Tag Key: **LeanSynthetics: Website Status**
- Tag Value: **UP** or **DOWN.**

In addition to storing the information in the Tag, the agent will also update:

- **System Instance Health**

That is, if the agent cannot find the specified keyword, then it will update the system instance health to UnplannedOutage. If the website is UP in the next sweep, then the system instance goes back to InOperation.

Tip! This agent can be easily updated to do more advanced tasks. For more information on how to do this refer to Python Splinter. Splinter is an open source tool for testing web applications using Python. It lets you automate browser actions, such as visiting URLs and interacting with their items.

Configuration

The agent configuration can be edited via config.ini

```
[APIDetails]
;Example API details - please replace with your information
app_id=EXAMPLE_ID
app_key=EXAMPLE_KEY
user_id=EXAMPLE_USER
enov8_url=https://example/ecosystem/
default_org=ECO-000000007898
```

```
;The amount of time (in hours) that must pass for the agent to be deemed InActive
failover_dur=24
```

API Details:	
app_id	Unique API ID that can be created in ecosystem by a system admin
app_key	Unique API Key is used with the above ID for authentication
user_id	Ecosystem User ID (This user will be the author of all changes)
enov8_url	The exact URL of your ecosystem instance
default_org	The system ID of the organisation that objects will be connected to
failover_dur	The amount of time (in hours) that must pass for the agent to be deemed InActive

The website configuration can be edited via an Integration object which is attached to a SystemInstance in ecosystem. This Integration object contains a JSON in the Value field which stores all fields below except for the Username and Password. These are stored as UserID and Password properties of the Integration object.

URL	The URL to open and search for the keyword
Keyword	The keyword to search for on the above website
Login	Whether the script should attempt to login to the given website. Must be True or False.
Username	The username used to login to the website if Login = True
Password	The password used to login to the website if Login = True
UserFieldID	The ID of the Username field on the given website
PassFieldID	The ID of the Password field on the given website
LoginButtonID	The ID of the Login button on the given website. If login = True.

Execution

Open Source Version: <my-python-path>/python.exe lean_synthetics.py

Binary Version: lean_synthetics.exe

Note: Once working you should schedule* agent to wake up & run periodically.

API Health

Introduction

This agent helps to determine if your APIs are healthy & performing. It will check the status code, content of response and response time for a specified API. If the result doesn’t meet the expected result, then the status of the corresponding system instance will be set to “UnplannedOutage”, otherwise it will be set to “InOperation” and tags will be created/updated.

How it Works

- For Each “System Instance”, use config to get
    - Target URL of API &
    - Expected Results (i.e. Status Code, Response Keyword, Response Time)

- Invoke API & fetch:
    - Status Code
    - Response (i.e. looks for expected keyword)
    - Response Time

- Updates Status
    - UnplannedOutage
    - InOperation

- Updates Tags of System Instance in EcoSystem
    - tag_key[“statuscode test“]
    - tag_key[“keyword test“]
    - tag_key[“response time test“]

Configuration

The agent configuration can be edited via config.ini

```
[APIDetails]
;Example API details - please replace with your information
app_id=EXAMPLE_ID
app_key=EXAMPLE_KEY
user_id=EXAMPLE_USER
enov8_url=http://example/ecosystem/
default_org=ECO-000000007898
;The amount of time (in hours) that must pass for the agent to be deemed InActive
failover_dur=24

;API test details - feel free to add more tests but keep 'API-Test-' as the prefix of the section (e.g.
API-Test-ecosystem)
[API-Test-1]
```

```
;Params to test
system_instance=GDW (SIT)
url=http://dummy.restapiexample.com/api/v1/employees
method=GET
param=
body=

;expected results - response time in ms
expected_statuscode=200
expected_keyword=example
expected_response_time=1000

[API-Test-2]
;Params to test
system_instance=SAP (SIT)
url=http://dummy.restapiexample.com/api/v1/create
method=POST
param=
body={"name":"uniqueName","salary":"123","age":"23"}

;expected results - response time in ms
expected_statuscode=201
expected_keyword=EXAMPLE
expected_response_time=1000
```

API Details:	
app_id	Unique API ID that can be created in ecosystem by a system admin
app_key	Unique API Key is used with the above ID for authentication
user_id	Ecosystem User ID (This user will be the author of all changes)
enov8_url	The exact URL of your ecosystem instance
default_org	The system ID of the organisation that objects will be connected to
failover_dur	The amount of time (in hours) that must pass for the agent to be deemed InActive

API-Test-[x]:	
system_instance	Resource name of the system instance
url	The URL of the API
method	HTTP method of the API
param	Parameters for the URL

body	Payload for POST method
expected_statuscode	Expected status code received from server
expected_keyword	Expected keyword in the response received from server
expected_response_time	Expected elapsed time of response for the API

Execution

Open Source Version: <my-python-path>/python.exe api_health.py

Binary Version: api_health.exe

Note: Once working you should schedule* agent to wake up & run periodically.

Discovery Agents

Component Discovery

Introduction

This agent uses NMAP to quickly find components / servers in your network. All servers / components on the network give of a “heat signature” or more precisely, one or more network signatures. This script uses that information to scan the network looking for signs of life & then add them to the Enov8 CMDB.

How it Works

- The agent reads the Config.ini to identify the network range to scan
  e.g. 127.0.0.05, 127.0.0.10 – 127.0.0.25.
- The agent scans the range looking for life (TCP/IP or UDP signature).
- New components are added to the Enov8 CMDB.

Configuration

The agent configuration can be edited via config.ini

```
[APIDetails]
;Example API details - please replace with your information
app_id=EXAMPLE_ID
app_key=EXAMPLE_KEY
user_id=EXAMPLE_USER
enov8_url=https://example/ecosystem/
default_org=ECO-000000007898
default_group=ECO-000000004182
;The amount of time (in hours) that must pass for the agent to be deemed InActive
failover_dur=24

[NETWORK]
;Example IP range - please replace with your information
IP = 172.36.68.06, 172.36.68.69, 172.36.68.25 - 172.36.68.36, 10.10.1.5, 172.16.0.4 - 172.16.0.9,
54.56.129.30
```

API Details:	
app_id	Unique API ID that can be created in ecosystem by a system admin
app_key	Unique API Key is used with the above ID for authentication
user_id	Ecosystem User ID (This user will be the author of all changes)

enov8_url	The exact URL of your ecosystem instance
default_org	The system ID of the organisation that objects will be connected to
default_group	The user ID assigned with the System Component
failover_dur	The amount of time (in hours) that must pass for the agent to be deemed InActive
Network:	
IP	The IP addresses of remote hosts to be scanned. Can include ranges and lists of IPs separated by commas

Execution

Open Source Version: <my-python-path>/python.exe component_discovery.py.

Binary Version: component_discovery.exe

Note: Once working you should schedule* agent to wake up & run periodically.

Service Discovery

Introduction

This agent is similar to component discovery, except it looks for specific network service ports. The agent checks to see if any remote hosts, with corresponding IP addresses & ports specified in the config.ini, exist. Clients can also instead scan ports of components in ecosystem.

Note: If a new resource is found then the component & port is added to the Enov8 EcoSystem Platform. The open ports found on the server are added to the system component as 'port monitors' and the result of the scan is added as a tag.

How it Works

- The agent reads the config to identify the network range to scan. e.g. 127.0.0.05, 127.0.0.10 – 127.0.0.25.
- Can be set to scan components in ecosystem instead by setting use_ip_range to false.
- The agent scans the range looking for life (TCP/IP or UDP signature).
- New components & service ports are added to the Enov8 CMDB.

Configuration

The agent configuration can be edited via config.ini

```
[APIDetails]
;Example API details - please replace with your information
app_id=EXAMPLE_ID
app_key=EXAMPLE_KEY
user_id=EXAMPLE_USER
enov8_url=https://example/ecosystem/
default_org=ECO-000000007898
default_group=ECO-000000004182
;The amount of time (in hours) that must pass for the agent to be deemed InActive
failover_dur=24

[NETWORK]
;Example IP ranges and Ports - please replace with your information
IP=172.36.68.26 - 172.36.68.36, 172.36.68.06, 172.36.68.69, 172.36.68.25
; if false, tool will use ecosystem components instead of ip range
use_ip_range=True
Port=80, 443, 1433, 3306
```

API Details:	
app_id	Unique API ID that can be created in ecosystem by a system admin
app_key	Unique API Key is used with the above ID for authentication

user_id	Ecosystem User ID (This user will be the author of all changes)
enov8_url	The exact URL of your ecosystem instance
default_org	The system ID of the organisation that objects will be connected to
default_group	The user ID assigned with the System Component
failover_dur	The amount of time (in hours) that must pass for the agent to be deemed InActive

Network:	
IP	The IP addresses of remote hosts to be scanned. Can include ranges and lists of IPs separated by commas
Port	The ports to be scanned for each IP address

Execution

Open Source Version: <my-python-path>/python.exe service_discovery.py

Binary Version: service_discovery.exe

Note: Once working you should schedule* agent to wake up & run periodically.

Other Agents

Trip Wire Agent

Introduction

The Tripwire agent helps you capture illegal change on a component.

Note: This agent needs to be installed locally on the target Component.

How it Works

- The agent reads the Config to identify Files or Folders to Monitor
- The agent uses a checksum to baseline the current contents
- If any of the Files or Folders change then a tag called “**tripwire: last modified**” is updated.

Configuration

The agent configuration can be edited via config.ini

```
[APIDetails]
;Example API details - please replace with your information
app_id=EXAMPLE_ID
app_key=EXAMPLE_KEY
user_id=EXAMPLE_USER
enov8_url=http://example/ecosystem/
default_org=ECO-000000007898
;The amount of time (in hours) that must pass for the agent to be deemed InActive
failover_dur=24

[machineDetails]
;The resource name of the corresponding system component in ecosystem
machineName=EXAMPLE_COMPONENT

[filesToCheck]
;The list of files to scan for changes. Format = 'identifier: file path'
example_file=C:\Users\example\test\file.txt
example_file_1=C:\Users\example\test\file1.txt
example_file_2=C:\Users\example\test\file2.txt
```

```
[foldersToCheck]
;The list of directories to scan for changes. Format = 'identifier: directory path'
example_folder=C:\Users\example\test
example_folder_1=C:\Users\example\test1
example_folder_2=C:\Users\example\test2
```

API Details:	
app_id	Unique API ID that can be created in ecosystem by a system admin
app_key	Unique API Key is used with the above ID for authentication
user_id	Ecosystem User ID (This user will be the author of all changes)
enov8_url	The exact URL of your ecosystem instance
default_org	The system ID of the organisation that objects will be connected to
failover_dur	The amount of time (in hours) that must pass for the agent to be deemed InActive

Machine Details:	
machineName	The name of a System Component in ecosystem – usually the machine this agent is currently being configured on

Files to Check:	
Unique ID for File	Exact file location to compare on this machine (e.g. /home/user/test/text.txt)

Folders to Check:	
Unique ID for Folder	Exact directory location to compare on this machine (e.g. /home/user/test)

Execution

Open Source Version: <my-python-path>/python.exe trip_wire.py.

Binary Version: trip_wire.exe

Note: Once working you should schedule* agent to wake up & run periodically.

Glances Agent

Introduction

An agent to capture resource specific insights like Process, Memory, CPU, Disk & Network Usage. Note: This agent requires a locally installed agent called “Glances” to be installed directly on the component (server) itself.

Note: This agent has a Dependency on PYPI Glances Agent being installed on target Component.

How it Works

Run Glances

Enov8 will poll the “Glances” Agent to check whether config conditions are met.

This includes:

- Process Monitor   Check processes identified in Config are running.
- Ram Monitor   Check Memory falls within “acceptable” range as identified in the Config.
- CPU Usage   Check CPU falls within “acceptable” range as identified in the Config.
- Disk Monitor   Check Disk usage within “acceptable” range as identified in the Config.
- Network Monitor   Check send/receive bitrate in “acceptable” range as identified in Config.

If config condition not met, then the SystemComponent status will be changed to UnplannedOutage.
If met, then status will be reflected as InOperation.

Configuration

The agent configuration can be edited via config.ini

```
;Example API details - please replace with your information
[APIDetails]
app_id = 1234
app_key = 1234
user_id = 1234
enov8_url = http://localhost/ecosystem_uat/
default_org = ECO-000000003945
;The amount of time (in hours) that must pass for the agent to be deemed
InActive
failover_dur = 24

;The details of the Glances API (localhost:61208) is the default port
[GlancesAPIDetails]
glances_url = http://localhost:61208/

;The resource name of the corresponding System Component in ecosystem to
monitor running processes
[machineDetails]
machineName = lnux101
```

```
;The list of process to check on this machine. Format =  'identifier: Full
Process name'
[processesToCheck]
Google Chrome = chrome.exe
Command Prompt = cmd.exe
Windows Explorer = explorer.exe
Outlook = outlook.exe
#test = test.exe

;The resource name of the corresponding eco class in ecosystem to monitor RAM
usage details
[resourceDetails]
resourceName = SAP (SVP2)
eco_class = systemInstance

[RAM]
;value in MB
min = 2000
max = 17000

;The resource name of the corresponding eco class in ecosystem to monitor the
CPU Usage
[CPU_resourceDetails]
CPU_resourceName = lean_synthetics
CPU_eco_class = systemInstance

[CPU_Process]
;value in percentage
min = 0
max = 75

;The resource name of the corresponding eco class in ecosystem to monitor the
Disk Usage
[disk_resourceDetails]
disk_resourceName = Salesforce (UAT3)
disk_eco_class = systemInstance
;value in MB
min = 5
max = 90

;The resource name of the corresponding eco class in ecosystem to monitor the
Network Usage
[network_resourceDetails]
network_resourceName = Salesforce (SIT4)
network_eco_class = systemInstance
;value in Byte per second
sent_min = 1
receive_min = 1
```

API Details	
app_id	Unique API ID that can be created in ecosystem by a system admin
app_key	Unique API Key is used with the above ID for authentication

user_id	Ecosystem User ID (This user will be the author of all changes)
enov8_url	The exact URL of your ecosystem instance
default_org	The system ID of the organisation that objects will be connected to
failover_dur	The amount of time (in hours) that must pass for the agent to be deemed InActive

GlancesAPIDetails	
machineName	The name of a System Component in ecosystem – usually the machine this agent is currently being configured on
Machine Details	
machineName	The name of a System Component in ecosystem.

Processes to Check	
processesToCheck	The names of the processes which needs to be check

Resource Details: This agent monitors the RAM usage details	
resourceName	The name of the System component or System Instance or System or Environment in ecosystem.
eco_class	The valid eco_class name could be systemComponent or systemInstance or system or environment
RAM	
min	Minimum limit of RAM in MB
Max	Maximum limit of RAM in MB

CPU Resource Details: This agent monitors the CPU usage details	
CPU_resourceName	The name of the System component or System Instance or System or Environment in ecosystem.
CPU_eco_class	The valid eco_class name could be systemComponent or systemInstance or system or environment
CPU Process	
min	Minimum limit of CPU in percentage
max	Maximum limit of CPU in percentage

Disk Resource Details: This agent monitors the Disk usage details	
disk_resourceName	The name of the System component or System Instance or System or Environment in ecosystem.
disk_eco_class	The valid eco_class name could be systemComponent or systemInstance or system or environment
min	Minimum limit of disk usage in MB
max	Maximum limit of disk usage in MB

Network Resource Details: This agent monitors the Network details	
network_resourceName	The name of the System component or System Instance or System or Environment in ecosystem.
network_eco_class	The valid eco_class name could be systemComponent or systemInstance or system or environment
sent_min	Minimum limit of data sent (Bytes/Sec)
receive_min	Minimum limit of data received (Bytes/Sec)

Execution

Open Source Version: <my-python-path>/python.exe glances_resource_monitor.py

Binary Version: glances_resource_monitor.exe

Note: Once working you should schedule* agent to wake up & run periodically.