DOKUMENT PROKAZUJÍCÍ DOSAŽENÍ VÝSLEDKU

FW10010040-V1: Monitorovací agent

Příloha k průběžné zprávě za rok 2024

Číslo projektu:	**FW10010040**
Název projektu:	**Kombinované pasivní a aktivní síťové monitorování**
Akronym:	**INVENTOR**
Účastníci projektu:	
Hlavní příjemce:	**Flowmon Networks a.s.**
Další účastník:	**Vysoké učení technické v Brně - Fakulta informačních technologií**

Obsah

Tento dokument popisuje dosažení výsledku V1 – Monitorovací agent v rámci projektu Kombinované pasivní a aktivní síťové monitorování (INVENTOR). Cílem monitorovacího agenta je umožnit vykonávání aktivního monitorování prostřednictvím odesílaní uměle vytvořeného síťového provozu do monitorované sítě a následné analyzování odpovědí na tato data. Součástí dokumentu je popis technické implementace spolu s přiloženou dokumentací. Dosažený výsledek má podobu samostatného software, který je veřejně dostupný prostřednictvím Git repositáře dostupného na adrese https://rysavy-ondrej.github.io/project-inventor/.

Požadavky

Před samotným návrhem a implementací jádra, bylo nutné specifikovat funkce, které bude muset výsledný software splňovat. V opačném případě by mohlo dojít k situaci, že některé funkce orchestrátoru nebude možné realizovat. Tyto požadavky jsou rozděleny do tří kategorií:

Požadavky na bezpečnost:

- zajištění, aby se výsledky testu dostaly pouze k oprávněným příjemcům,
- přístup k testům rozdělit na dva typy – plný přístup s možností provádění úprav testu a přístup pouze pro čtení,
- zajištění bezpečnosti vůči útokům – nespoléhat se pouze na šifrovaní TLS,
- AAA – autentizace, autorizace, audit – každá API transakce se eviduje.

Požadavky na odolnost:

- chyba v jednom testu nesmí ovlivnit běh ostatních agentů nebo celého agenta,
- na stroji nemohou běžet neukončené procesy v případě chyb u testů,
- musí být implementován mechanismus zabraňující zaplnění disku pomocí mazání starých dat,
- rozšířené možnosti logování – syslog, logstash, splunk.

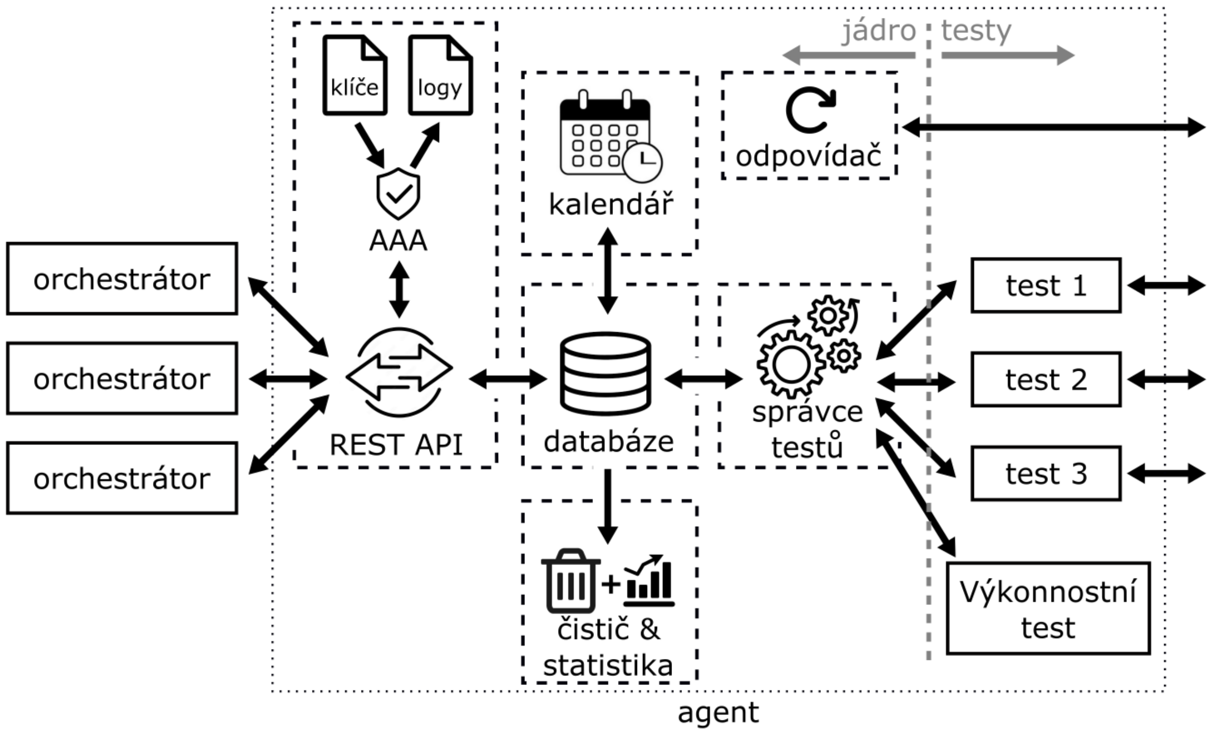
Požadavky na funkcionalitu:

- podpora komunikace s více orchestrátory najednou,
- vytvořené testy je možné upravovat za běhu bez nutnosti restartování,
- sdílení výsledků jednoho testu mezi více orchestrátory,
- agent musí podporovat jak jednorázové testy, tak pravidelně se spouštějící,
- agent musí být dostupný skrze rozhraní REST API,
- modulární architektura umožňující jednoduché přidávání testů,
- možnost vzdáleně sledovat vytížení stroje na kterém běží agent,
- možnost nasadit agenta i mimo řešení Flowmon,
- možnost přenášet logy od agenta do orchestrátoru,

- již naplánované testy je možné za běhu upravovat,
- agent musí být schopen reagovat na test, který selhal, formou vytvoření dodatečného testu,
- možnost stahovat výsledky z vícero testů najednou.

Architektura

Na základě specifikovaných požadavků byla vytvořena architektura, která je zobrazená na obrázku 1.



Obrázek 1: Architektura agenta.

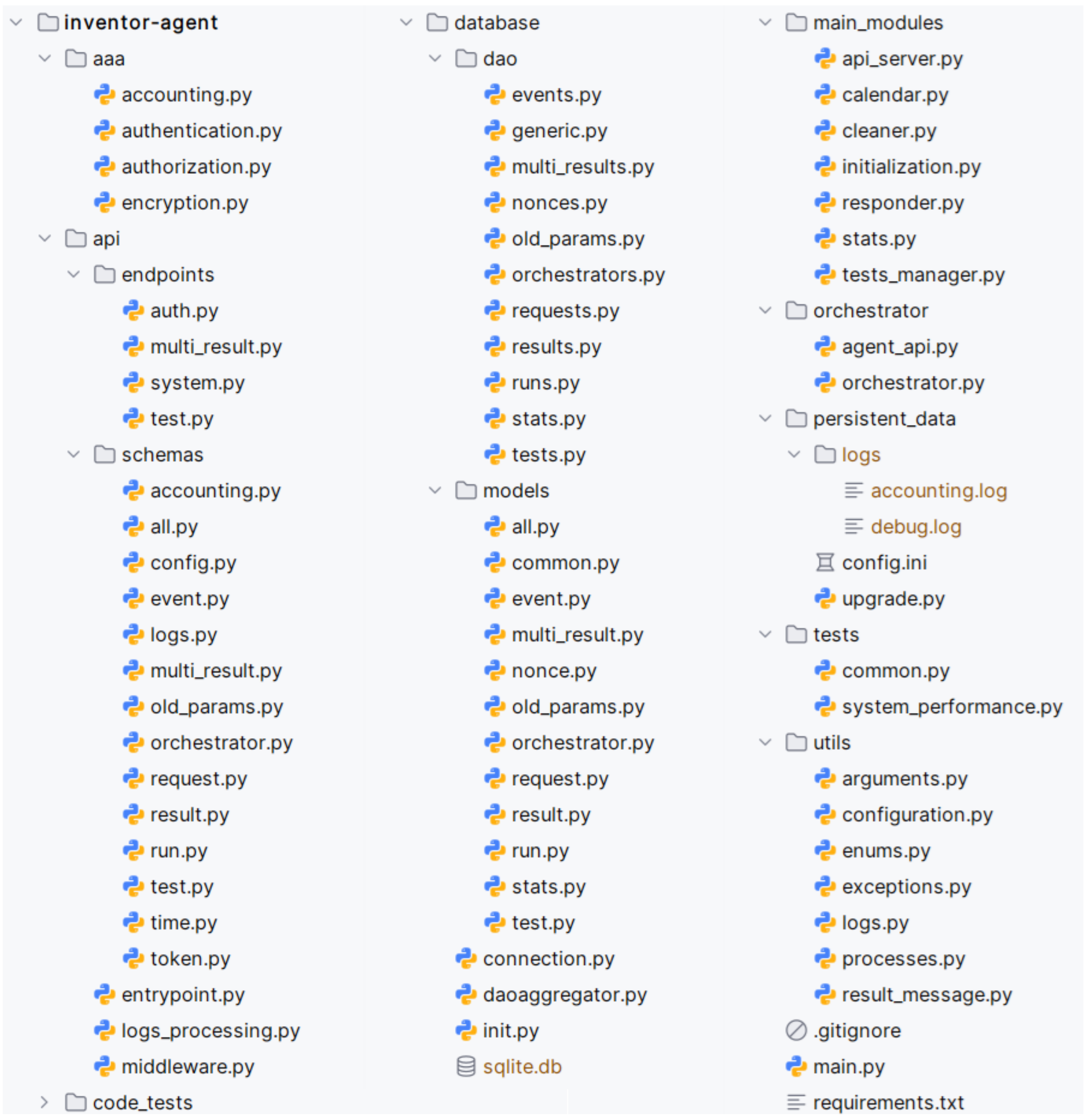
Architektura se skládá z několika bloků:

- **Orchestrátor** – Orchestrátor slouží pro konfiguraci agenta – specifikuje testy, které mají běžet a následně shromažďuje výsledky z těchto testů.
- **Rozhraní REST API** – Jedná se o rozhraní pro komunikaci mezi agentem a orchestrátory. Na agentovi bude běžet nonstop server, který bude očekávat příchozí HTTP požadavky, na které bude vracet odpovědi.
- **AAA** – Služba pro autentizaci, autorizaci a audit každého příchozího požadavku na REST API.
- **Klíče** – Pokud nejsou příchozí požadavky spojeny s konkrétním testem, jsou požadavky autorizovány podle klíčů uložených v souboru.

- **Logy** – Jedná se o soubory, do kterých se ukládají aplikační logy (typicky oznamující neočekávané situace při běhu programu) nebo transakční logy (zaznamenávající všechny zpracované transakce).
- **Databáze** – Persistentní relační úložiště typu SQL databáze. Obsahuje několik tabulek, do kterých přistupují jednotlivé procesy agenta.
- **Kalendář** – Slouží pro spuštění testů, zaručuje, že testy jsou prováděny v pravidelných intervalech.
- **Správce testů** – Je zodpovědný za spuštění testu, čekání na příjem výsledku a ukončení všech testů v požadovaném čase.
- **Testy** – Jedná se o testy, které jsou zodpovědné za monitorování konkrétních aplikací a služeb. Každý test je zodpovědný právě za jednu službu.
- **Výkonnostní test** – Specifický test, který neslouží pro monitorování vzdálených služeb, ale slouží pro zjišťování vytížení zařízení, na kterém běží agent.
- **Odpovídač** – Dalším specifickým testem je test zaměřený na monitorování linek mezi dvěma agenty. Odpovídač slouží k tomu, aby odpovídal na požadavky od jiných agentů.
- **Čistič a statistika** – Jedná se o dva procesy, které jsou spouštěny v pravidelných intervalech. Cílem je ukládat statistické informace o aktuální konfiguraci agenta a odstranění starých záznamů ze všech tabulek v databázi.

Popis zdrojového kódu

Jádro agenta je implementováno v jazyce Python 3.11, což umožňuje jeho spouštění na různých platformách (Windows, Linux, MacOS) a architekturách (x86, x64, ARM). Strukturu projektu a jednotlivé zdrojové soubory je možné vidět na obrázku 2. Popis se nachází v následujícím textu.



```
inventor-agent
    aaa
        accounting.py
        authentication.py
        authorization.py
        encryption.py
    api
        endpoints
            auth.py
            multi_result.py
            system.py
            test.py
        schemas
            accounting.py
            all.py
            config.py
            event.py
            logs.py
            multi_result.py
            old_params.py
            orchestrator.py
            request.py
            result.py
            run.py
            test.py
            time.py
            token.py
        entrypoint.py
        logs_processing.py
        middleware.py
    code_tests
    database
        dao
            events.py
            generic.py
            multi_results.py
            nonces.py
            old_params.py
            orchestrators.py
            requests.py
            results.py
            runs.py
            stats.py
            tests.py
        models
            all.py
            common.py
            event.py
            multi_result.py
            nonce.py
            old_params.py
            orchestrator.py
            request.py
            result.py
            run.py
            stats.py
            test.py
        connection.py
        daoaggregator.py
        init.py
        sqlite.db
    main_modules
        api_server.py
        calendar.py
        cleaner.py
        initialization.py
        responder.py
        stats.py
        tests_manager.py
    orchestrator
        agent_api.py
        orchestrator.py
    persistent_data
        logs
            accounting.log
            debug.log
        config.ini
        upgrade.py
    tests
        common.py
        system_performance.py
    utils
        arguments.py
        configuration.py
        enums.py
        exceptions.py
        logs.py
        processes.py
        result_message.py
    .gitignore
    main.py
    requirements.txt
```

Obrázek 2: Struktura zdrojových kódů.

Seznam souborů ve složce “**aaa/**”:

Soubor	Popis souboru a poskytovaných funkcí
accounting.py	Soubor otevírá spojení s lokálním souborem a poskytuje rozhraní pro zápis účetních (jinými slovy také transakčních nebo audit) záznamů.
authentication.py	Vytváří autentizační token pro orchestrátory, který obsahuje šifrovaná data nutná pro autentizaci požadavků. Také poskytuje funkci pro dešifrování dat z tokenu a kontrolu autentizace.
authorization.py	Poskytuje funkci pro kontrolu, zda je požadavek oprávněn přistupovat k zadanému testu. Řeší kontrolu časové platnosti

	požadavku, jeho unikátnosti a porovnává správný podpis požadavku.
encryption.py	Poskytuje funkce potřebné pro šifrování, dešifrování dat a pro výpočet kontrolních otisků (hash).

Seznam souborů ve složce “**api/endpoints/**”:

Soubor	Popis souboru a poskytovaných funkcí
endpoints/auth.py	Definuje REST API koncové body (endpoints) pro autentizaci orchestrátoru.
endpoints/multi_results.py	Definuje REST API koncové body (endpoints) pro práci s výsledky několika testů najednou.
endpoints/system.py	Definuje REST API koncové body (endpoints) pro práci s konfigurací a logovacími soubory.
endpoints/test.py	Definuje REST API koncové body (endpoints) pro práci s testy.
schemas/*.py	Každý soubor obsahuje specifikaci datového modelu určující požadované atributy požadavků a odpovědí rozhraní REST API.
entrypoint.py	Specifikace REST API serveru založeného na knihovně FastAPI.
logs_processing.py	Poskytuje funkce pro práci s logovacími soubory – výběr dat, výpočet statistik, komprese dat.
middleware.py	Definuje funkce, které se mají spouštět pro každý přijatý požadavek na REST API. Konkrétně se jedná o autentizaci a audit.

Seznam souborů ve složce “**code_tests/**”:

Soubor	Popis souboru a poskytovaných funkcí
***.py**	Složka obsahuje několik podsložek obsahující soubory definující jednotkové (unit) testy pro různé zdrojové kódy.

Seznam souborů ve složce “**database/**”:

Soubor	Popis souboru a poskytovaných funkcí
dao/*.py	Složka obsahuje funkce pro přístup k datům uloženým v příslušných tabulkách – Data Access Object (DAO) funkce.
models/*.py	Složka obsahuje soubory definující datovou strukturu všech SQL tabulek.
connection.py	Slouží pro navázání spojení s SQL databází.
daoaggregator.py	Funkce obsahuje třídu, která agreguje všechny DAO funkce ze složky “database/dao/” a umožňuje tak jednotné rozhraní pro přístup do SQL databáze.

init.py	Soubor vytvářející prvotní data umožňující okamžitou práci s agentem a vzorovými daty.
sqlite.db	Databázový soubor obsahující kompletní SQL databázi – tabulky a data.

Seznam souborů ve složce “**main_modules/**”:

Soubor	Popis souboru a poskytovaných funkcí
api_server.py	Obsahuje funce pro inicializaci a spuštění REST API server.
calendar.py	Definuje funkce pro práce s kalendářem.
cleaner.py	Obsahuje funkce pro mazání starých záznamů v SQL tabulkách.
initialization.py	Inicializační soubor, který vytváří proměnné konfiguračního souboru a kontroluje správnost databáze.
responder.py	Obsahuje kód pro spuštění UDP serveru a pro odesílání odpovědí na přijaté dotazy.
stats.py	Definuje funkce pro počítání statistik jednotlivých SQL tabulek a ukládání těchto výsledků.
tests_manager.py	Specifikuje funkce pro spouštění testů, čekání na výsledky od testů a pro ukončení testů.

Seznam souborů ve složce “**orchestrator/**”:

Soubor	Popis souboru a poskytovaných funkcí
api_server.py	Poskytuje základní funkce pro komunikaci mezi orchestrátorem a agentem. Tyto funkce také zahrnují autentizaci.
orchestrator.py	Implementuje rozhraní mezi orchestrátorem a agentem umožňující vytvářet testy, stahovat data a volat všechny definované REST API koncové body.

Seznam souborů ve složce “**persistent_data/**”:

Soubor	Popis souboru a poskytovaných funkcí
logs/accounting.log	Soubor obsahující programové logovací záznamy.
logs/debug.log	Soubor obsahující transakční (audit) záznamy.
config.ini	Konfigurační soubor ve formátu YAML obsahující parametry pro jednotlivé části programu.
upgrade.py	Soubor pro aktualizaci programu.

Seznam souborů ve složce “**tests/**”:

Soubor	Popis souboru a poskytovaných funkcí

common.py	Definuje funkce pro práci s meziprocesorovou frontou pro výměnu dat mezi testu a jádrem agenty.
system_performance.py	Speciální test, který monitoruje vytížení stávajícího stroje.

Seznam souborů ve složce “**utils/**”:

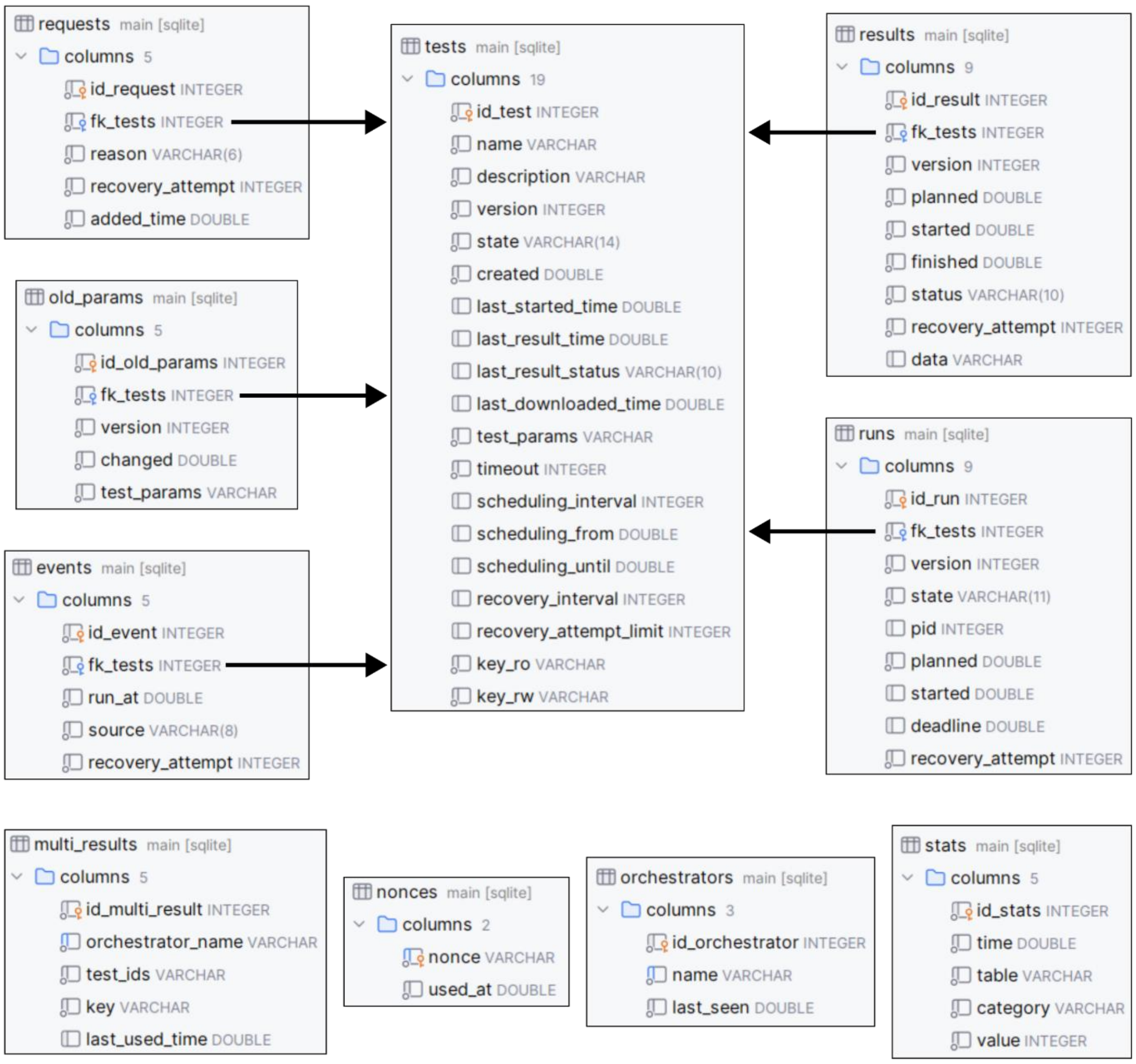
Soubor	Popis souboru a poskytovaných funkcí
arguments.py	Soubor obsahuje funkce pro zpracování argumentů příkazového řádku.
configuration.py	Poskytuje funkce pro práci s konfiguračním souborem jako čtení nebo zapisování hodnot.
enums.py	Obsahuje definici datových typů typu enum (množiny hodnot), které jsou využívány v celém programu.
exceptions.py	Definuje výjimky, které je možné v průběhu programu vytvořit.
logs.py	Soubor otevírá spojení s lokálním souborem a poskytuje rozhraní pro zápis programových záznamů různých úrovní (debug až critical).
processes.py	Specifikuje funkce pro spuštění a ukončení systémových procesů.
result_message.py	Definuje datový typ, do kterého jsou ukládány výsledky z jednotlivých testů.

Seznam souborů ve složce “**/**”:

Soubor	Popis souboru a poskytovaných funkcí
main.py	Vstupní bod programu, který spouští příslušnou funkci dle zadaného parametru.
requirements.txt	Seznam Python balíčků, na kterých je jádro agenta závislé.

Databázový systém

Pro správný běh programu vyžadují jednotlivé procesy práci s daty, která musejí být persistentně uložena na agentovi. K tomu byla zvolena relační SQL databáze. Následující obrázek 3 zobrazuje schéma SQL tabulek.



requests main [sqlite]
columns 5

id_request	INTEGER
fk_tests	INTEGER
reason	VARCHAR(6)
recovery_attempt	INTEGER
added_time	DOUBLE

tests main [sqlite]
columns 19

id_test	INTEGER
name	VARCHAR
description	VARCHAR
version	INTEGER
state	VARCHAR(14)
created	DOUBLE
last_started_time	DOUBLE
last_result_time	DOUBLE
last_result_status	VARCHAR(10)
last_downloaded_time	DOUBLE
test_params	VARCHAR
timeout	INTEGER
scheduling_interval	INTEGER
scheduling_from	DOUBLE
scheduling_until	DOUBLE
recovery_interval	INTEGER
recovery_attempt_limit	INTEGER
key_ro	VARCHAR
key_rw	VARCHAR

results main [sqlite]
columns 9

id_result	INTEGER
fk_tests	INTEGER
version	INTEGER
planned	DOUBLE
started	DOUBLE
finished	DOUBLE
status	VARCHAR(10)
recovery_attempt	INTEGER
data	VARCHAR

old_params main [sqlite]
columns 5

id_old_params	INTEGER
fk_tests	INTEGER
version	INTEGER
changed	DOUBLE
test_params	VARCHAR

runs main [sqlite]
columns 9

id_run	INTEGER
fk_tests	INTEGER
version	INTEGER
state	VARCHAR(11)
pid	INTEGER
planned	DOUBLE
started	DOUBLE
deadline	DOUBLE
recovery_attempt	INTEGER

events main [sqlite]
columns 5

id_event	INTEGER
fk_tests	INTEGER
run_at	DOUBLE
source	VARCHAR(8)
recovery_attempt	INTEGER

multi_results main [sqlite]
columns 5

id_multi_result	INTEGER
orchestrator_name	VARCHAR
test_ids	VARCHAR
key	VARCHAR
last_used_time	DOUBLE

nonces main [sqlite]
columns 2

nonce	VARCHAR
used_at	DOUBLE

orchestrators main [sqlite]
columns 3

id_orchestrator	INTEGER
name	VARCHAR
last_seen	DOUBLE

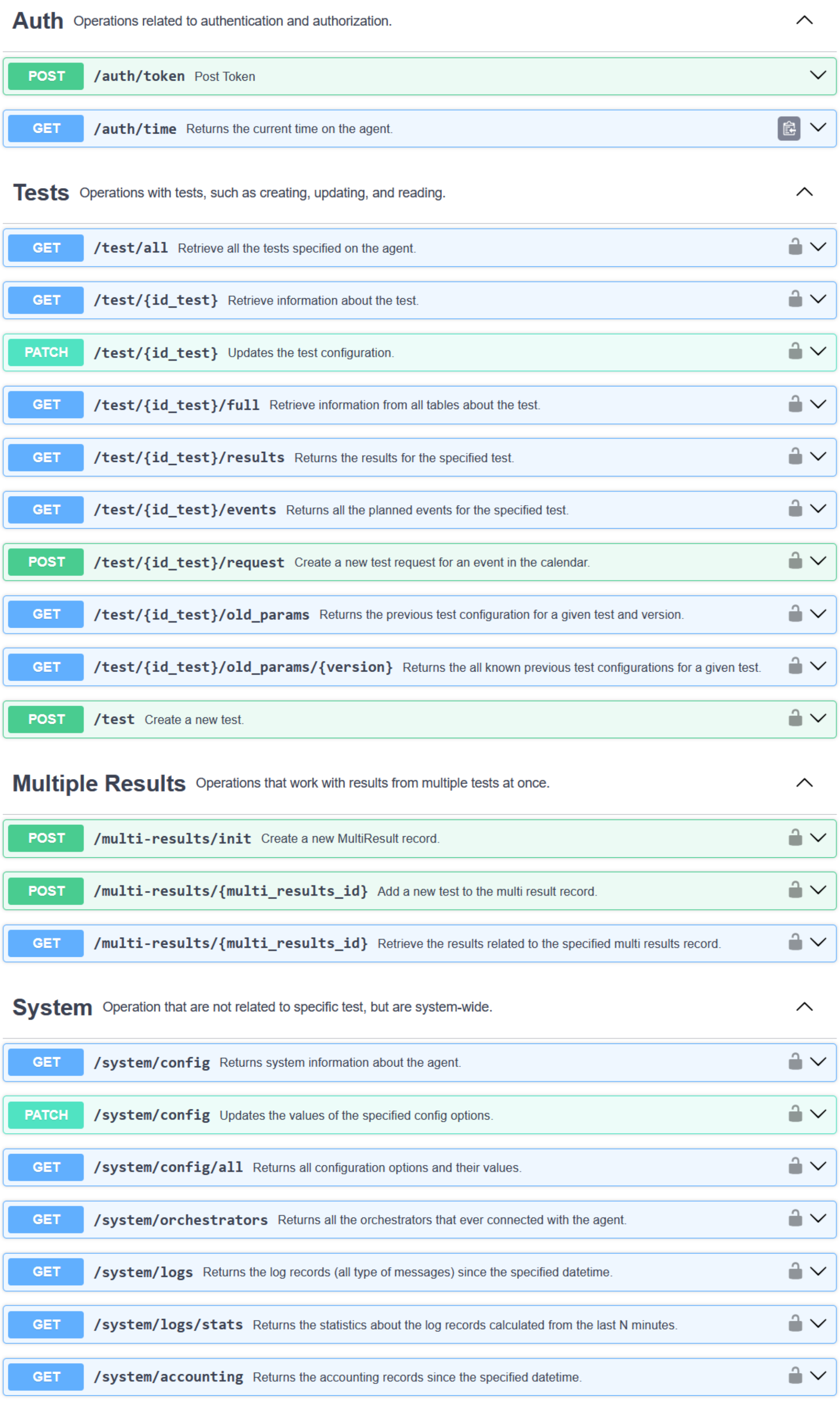
stats main [sqlite]
columns 5

id_stats	INTEGER
time	DOUBLE
table	VARCHAR
category	VARCHAR
value	INTEGER

Obrázek 3: Databázový SQL model.

REST API

Pro implementaci orchestrátoru, nebo jiného nástroje komunikujícího s agentem, bylo na agentovi vytvořeno REST API rozhraní. Rozhraní je definováno pomocí koncových bodů (endpoints), které definují formát vstupních a výstupních dat pro jednotlivé funkce rozhraní. Seznam bodů spolu se stručným popisem je zobrazen na obrázku 4. Plný popis je přiložen na konci této zprávy (v příloze).



Auth Operations related to authentication and authorization.

POST /auth/token Post Token

GET /auth/time Returns the current time on the agent.

Tests Operations with tests, such as creating, updating, and reading.

GET /test/all Retrieve all the tests specified on the agent.

GET /test/{id_test} Retrieve information about the test.

PATCH /test/{id_test} Updates the test configuration.

GET /test/{id_test}/full Retrieve information from all tables about the test.

GET /test/{id_test}/results Returns the results for the specified test.

GET /test/{id_test}/events Returns all the planned events for the specified test.

POST /test/{id_test}/request Create a new test request for an event in the calendar.

GET /test/{id_test}/old_params Returns the previous test configuration for a given test and version.

GET /test/{id_test}/old_params/{version} Returns the all known previous test configurations for a given test.

POST /test Create a new test.

Multiple Results Operations that work with results from multiple tests at once.

POST /multi-results/init Create a new MultiResult record.

POST /multi-results/{multi_results_id} Add a new test to the multi result record.

GET /multi-results/{multi_results_id} Retrieve the results related to the specified multi results record.

System Operation that are not related to specific test, but are system-wide.

GET /system/config Returns system information about the agent.

PATCH /system/config Updates the values of the specified config options.

GET /system/config/all Returns all configuration options and their values.

GET /system/orchestrators Returns all the orchestrators that ever connected with the agent.

GET /system/logs Returns the log records (all type of messages) since the specified datetime.

GET /system/logs/stats Returns the statistics about the log records calculated from the last N minutes.

GET /system/accounting Returns the accounting records since the specified datetime.

Obrázek 4: Seznam implementovaných REST API koncových bodů.

Spuštění agenta

Samotný běh agenta je složen z několika nezávislých procesů, které jsou potřeba pro správný běh programu. Každý proces plní jednu specifickou úlohu a je nutné ho spustit nezávisle. K tomu slouží soubor main.py, který je spouští následujícím příkazem:

```
python3 main.py --task <název_úlohy> --persistent
<cesta_k_složce>
```

Název úlohy pak může nabývat jednu z těchto hodnot:

- **init_database** – Jednorázový proces pro vymazání databáze a vytvoření všech potřebných SQL tabulek.
- **calendar** – Proces, který v nekonečném cyklu kontroluje požadavky pro spuštění nových testů a aktuálně naplánované testy. Požadavky na nové testy jsou uloženy v tabulce "request". Tato tabulka obsahuje buď žádosti o spuštění úplně nových testů, žádost o opakované spuštění testu z důvodu jeho selhání nebo odstranění naplánovaného testu při vypnutí nebo úplném odstranění testu. Při požadavku o nové spuštění se do tabulky "events" vytvoří nový záznam specifikující, který test a v kterou dobu se má spustit.
  Kontrola naplánovaných testů spočívá v neustálém dotazování na tabulku "events", zda se v ní nenachází záznamy, u kterých bylo dosaženo času spuštění. Takové záznamy jsou z tabulky odstraněny, jsou vytvořeny požadavky na spuštění nového procesu skrze tabulku "runs" a případně je vytvořen nový záznam do tabulky "events" specifikující čas dalšího spuštění v budoucnu.
- **cleaner** – Proces v pravidelných intervalech (ve výchozím nastavení 1x za hodinu) kontroluje obsah všech tabulek a vymazává z nich záznamy starší než vypočtený časový limit. U některých tabulek se může jednat o záznamy, které z důvodu chyby nebyly odstraněny očekávaným způsobem, u jiných se zase jedná o jediný způsob, jak jsou záznamy z tabulek mazány (například tabulka uložených "nonces" nebo výsledků z testů).
- **responder** – Jedná se o proces, který poslouchá na zadaném UDP portu, na kterém očekává zprávy z testu monitorujícího linky mezi dvěma agenty. Proces na přijaté zprávy reaguje odesláním odpovědi zpět.
- **server** – REST API server, který přijímá data od orchestrátorů a reaguje na ně buď vrácením dat z tabulek a lokálních souborů (konfigurace, logy) nebo je výsledkem zápis dat do tabulek nebo úprava konfiguračního souboru. Pro svou činnost server také využívá tabulky "nonces" a "orchestrators". Tabulka "nonces" slouží

pro uložení náhodně vygenerovaných nonces. To umožňuje, že server neakceptuje dva požadavky se stejnou hodnotu. Tabulka s orchestrátory slouží pouze pro evidenci, s kterými orchestrátory agent komunikuje a kdy proběhla poslední komunikace.

- **stats** – Jedná se o pomocný proces, který slouží pouze k diagnostickým účelům. Proces v pravidelných intervalech počítá množství záznamů dle typů záznamů v jednotlivých tabulkách. Tyto informace mohou v budoucnu pomoct s identifikováním chyb, kdy bude některých záznamů uložené neočekávaně velké nebo malé množství. Statistická data jsou uložena do tabulky “stats”.
- **tests_manager** – Správce procesů, který také běží v nekonečné smyčce a má za úkol tři činnosti: spouštení nových procesů (testů), sbírání výsledků z procesů a sledování správnosti ukončení procesů. Nové procesy se spouštějí na základě záznamů v tabulce “runs”. Výsledky z procesů se pak ukládají do tabulky “results”. V případě, když některý z procesů běží příliš dlouho (podle konfigurace), je proces násilně ukončen a na výsledky se již nečeká.

Monitorovací testy

Monitorovací agent na základě požadované konfigurace spouští jednotlivé testovací procedury, které jsou implementované jako Python skripty. Každý tento skript získává od agenta vstupní JSON konfiguraci požadovaného testu a po provedení testu generuje popis výsledku formou JSON výstupu.

Testovací případy jsou rozděleny do kategorií podle jejich zaměření:

- **Network**: Tato kategorie zahrnuje testy pro hodnocení základních a komplexních funkcí síťových služeb, včetně diagnostiky (ping, traceroute), dostupnosti služeb (DNS, SMTP, IMAP), účinnosti protokolů (MQTT, NTP) a aktivního monitorování (SNMP). Tyto testy jsou nezbytné pro ověření síťové konektivity, výkonu a funkčních aspektů síťové komunikace.
- **Performance**: Testy v této kategorii se zaměřují na výkonnostní aspekty síťové komunikace, jako je doba navázání spojení (TCP handshake), měření doby odezvy (doba do prvního bajtu) a propustnost (celkový počet stažených bajtů, měření šířky pásma). Tyto testy jsou klíčové pro vyhodnocení efektivity a rychlosti sítě.
- **Webapp**: Tato kategorie zahrnuje testy související s webovými aplikacemi, kontrolu protokolů (WebSocket, HTTP), simulaci interakce uživatele (přehrávání relace), analýzu načítání stránek (vodopád HTML) a doručování obsahu (snímky obrazovky webových stránek). Zahrnuje také testování rozhraní API (REST API), které zajišťuje správnou funkčnost a strukturu odpovědí.
- **Security**: Cílem testů v této kategorii je zajistit bezpečnost a integritu síťové komunikace a služeb. Patří sem testování protokolů (SSH, SSL/TLS), provádění hodnocení zabezpečení (bezpečnostní testy RapidSpike) a hodnocení adresářových služeb (LDAP).
- **Other**: Patří sem specializované testy, jako je gRPC pro efektivní komunikaci služeb, a testování databázových operací pro databáze SQL (PostgreSQL, MSSQL, MySQL) i NoSQL, které zajišťují výkonnost a spolehlivost databází.

Každá kategorie je navržena tak, aby se zabývala specifickými aspekty funkčnosti sítě a služeb a přispívala ke komplexnímu hodnocení celkového stavu a výkonu systému.

Seznam implementovaných testů

Implementované testy jsou uvedeny v následující tabulce:

Kategorie	Název testu	Popis testu
Network	network.ping	Testuje dosažitelnost hostitele na IP síti a měří dobu zpáteční cesty (RTT) zpráv.

Network	network.traceroute	Sleduje cestu paketů zdroje do určeného cíle v počítačové síti a identifikuje každý mezistupeň na trase.
Network	network.dns.resolve	Ověřuje funkčnost DNS služeb pomocí standardních metod a sbírá výsledky.
Network	network.smtp	Testuje schopnost odesílání e-mailů pomocí protokolu SMTP.
Network	network.imap	Testuje schopnost načítání e-mailů pomocí protokolu IMAP.
Network	network.mqtt	Testuje protokol MQTT pro lehké publikování a odběr zpráv.
Network	network.ntp	Testuje protokol pro synchronizaci času v počítačových systémech.
Network	network.snmp	Aktivní monitorování vybraných SNMP objektů (využití CPU, paměti, diskového prostoru, počet záznamů v tabulce atd.)
Network	network.ftp	Testuje protokol pro přenos souborů mezi klientem a serverem v počítačové síti.
Performance	performance.bandwidth	Měří síťové výkonnostní metriky jako jsou jitter, latence, ztráta paketů a propustnost mezi agenty.
Webapp	webapp.http	Testuje Hypertext Transfer Protocol pro komunikaci mezi webovými servery a klienty.
Webapp	webapp.security	Testuje bezpečnost koncového HTTP uzlu s ohledem na známé CVE zranitelnosti.
Webapp	Webapp.rest	Testuje RESTful API pro správnou odezvu a strukturu, zaměřuje se na analýzu JSON objektů.
Security	security.ssh	Testuje protokol Secure Shell pro zabezpečené síťové služby přes nezabezpečenou síť.
Security	security.tls	Testuje protokoly SSL/TLS pro zabezpečení šifrovaných spojení mezi síťově propojenými počítači.
Security	security.ldap	Testuje protokol LDAP.

Other	other.sql	Testuje databázové operace pro relační databáze, jako jsou PostgreSQL, MSSQL a MySQL.
Other	other.nosql	Testuje databázové operace pro vybrané NoSQL databáze.

Implementace testu

Každý test v rámci systému je implementován jako samostatný modul napsaný v jazyce Python a to ve standardizované struktuře. Jednotlivé testy jsou organizovány do adresářů, jejichž názvy odpovídají formátu "category.testname", což značí kategorii a specifický název testu. Každý test musí obsahovat minimálně následující soubory:

- **Hlavní skript:** Každý testovací modul obsahuje hlavní Python skript pojmenovaný "category_testname.py".
- **requirements.txt:** Každý adresář testu obsahuje soubor requirements.txt, který definuje externí závislosti potřebné pro běh testu, jako jsou knihovny a moduly, které musí být nainstalovány.
- **Readme.md:** Pro každý test je připraven soubor Readme.md, který poskytuje uživatelům a vývojářům podrobný popis testu, včetně formátu vstupní konfigurace a očekávaného formátu výstupních reportů. Dále zahrnuje příklady vstupních konfigurací pro ověření funkčnosti testů.

Jednoduché testy v systému jsou plně implementovány v jazyce Python, což umožňuje vysokou míru flexibility a přenositelnosti. Tyto testy využívají buď standardní knihovny Pythonu nebo knihovny třetích stran, které rozšiřují možnosti základních funkcí, například network.ping. Tento test využívá knihovnu icmplib pro odesílání a příjem ICMP echo žádostí a odpovědí. Ping test je základní diagnostický nástroj pro ověření dostupnosti hostitele na IP síti a měření doby odezvy (RTT). Tyto testy jsou celé realizovány v jazyce Python a nevyžadují spouštění externích aplikací, což znamená, že nejsou závislé na specifické systémové konfiguraci nebo externích nástrojích, snadno se konfigurují a spouštějí, a jejich výsledky jsou přímočaře interpretovatelné.

Na druhou stranu, složitější testy mohou vyžadovat interakci s externími aplikacemi, které poskytují rozšířené diagnostické možnosti nebo umožňují testování specifických aspektů systémové bezpečnosti a výkonnosti. Příklad složitého testu je webapp.security. Tento test používá shell skript testssl.sh pro provádění hloubkového bezpečnostního testování HTTPS serverů. Skript testssl.sh je robustní nástroj napsaný pro shell, který pomocí různých systémových nástrojů testuje bezpečnostní nastavení SSL/TLS protokolů na serverech. Výstup z tohoto skriptu je následně čten a interpretován Python skriptem testu, který převádí data do unifikovaného formátu JSON. Tento JSON výstup je posléze k dispozici pro další analýzu nebo reportování v rámci testovacího frameworku.

Pro implementaci testů je možné použít šablonu uvedenou níže. Tato šablona ukazuje, že test je odvozen z bázové třídy BaseTest, která poskytuje základní mechanismy pro vytvoření testu, zpracování vytvořené zprávy a definuje abstraktní metody, které je nutné v každém testu implementovat. Při inicializaci instance třídy Test je důležité předat jí frontu (Queue), která se používá pro komunikaci s monitorovacím agentem a pro ukládání výsledků testu. Tato fronta je předána nadřazené třídě pomocí metody init, která je volána v konstruktoru s parametrem queue.

Hlavní funkcionalitu třídy zajišťuje metoda **run**, která přijímá slovník s parametry potřebnými pro test (params) a identifikátor běhu testu (run_id). Během provádění testu může dojít k výjimkám, které jsou ošetřeny v bloku except. V případě výjimky metoda nastaví výsledek testu na selhání a může do výstupní zprávy message přidat detailní informace o chybě. Nezávisle na výsledku testu se ve finálním bloku finally vždy volá metoda process_message, která zpracuje zprávu message obsahující výsledky testu. Tato metoda implicitně přidá zprávu do fronty předané při inicializaci testu.

Kromě toho třída Test obsahuje statickou metodu deadline_calculation, která na základě vstupních parametrů vypočítá maximální povolenou dobu trvání testu (timeout). Tato hodnota timeoutu je klíčová pro správné nastavení a omezení doby běhu testů, aby nedošlo k jejich nekonečnému běhu v případě problémů v testovaném systému.

```
from multiprocessing import Queue
from common import BaseTest

class Test(BaseTest):

  def init(self, queue: Queue): super().init(queue)

  def run(self, params: dict, run_id: int) -> None:
    message = None
    try:

    “””
    Implement the regular test here,
    the result should be provided in message dictionary
    “””

    except:

    “””
    Implement exception:
       set the result of the test to fail
       optionally add detailed information to message
    """
    finally:
        self.process_message(message)
        return message


  def deadline_calculation(params: dict):

    """
    Calculate timeout here
    """
    return timeout
```

Příklad implementace a použití testu

Pro ilustraci principů implementace testů bude použit jednoduchý test network.icmp. Tento test slouží k diagnostice sítě pomocí ICMP (Internet Control Message Protocol). Test je používán k ověření dostupnosti hostitele na IP síti a měření doby zpáteční cesty (RTT) zpráv odeslaných z původního hostitele na cílový počítač.

Test je celý implementován v souboru network_ping.py. Ten obsahuje tyto hlavní části:

- Import potřebných knihoven jako icmplib, která umožňuje práci s ICMP zprávami.
- Slovník pro mapování ICMP typů zpráv (jejich kódů) na textové popisy.
- Funkce pro načtení a zpracování konfigurace, ověření její správnosti.
- Hlavní funkci run, která slouží jako hlavní vstupní bod pro spouštění testu s možností výstupu do fronty pro asynchronní zpracování.
- Implementaci ping testu pomocí knihovny icmplib.

Tato struktura je typická pro většinu testů. Test je konfigurován pomocí JSON souboru, který specifikuje cílový hostitel, velikost ICMP zátěže, počet ICMP paketů, prodlevu mezi pakety a maximální časový limit pro odpověď. Příklad vstupní konfigurace:

```
{
    "target_host": "8.8.8.8",
    "packet_size": 200,
    "packet_count": 2,
    "interpacket_delay": 3,
    "timeout": 5
}
```

Výsledky jsou vráceny ve formátu JSON, který obsahuje informace o běhu testu, včetně počtu odeslaných a přijatých paketů, ztrát, minimální, maximální, průměrné RTT, směrodatné odchylce RTT a jitteru. Výstup zahrnuje také detaily o každém ICMP echo requestu a odpovědi, včetně časových známek, RTT a stavu.

Příkladem úspěšného testu je výpis níže. Výsledek testu Network Ping s identifikačním číslem běhu run_id = 1 ukazuje, že test byl úspěšně dokončen (status = "completed"). Testování bylo zaměřeno na hostitele s IP adresou 8.8.8.8. Výsledky tohoto testu ukazují, že síťová komunikace s cílovou IP adresou je stabilní s plným přijetím všech testovacích paketů. Velký rozdíl mezi minimálním a maximálním RTT a vysoký jitter poukazují na možnou nestabilitu v síťovém provozu nebo v kolísání kvality spojení v čase testování. Přestože oba pakety byly úspěšně přijaty, rozdíl v časech odezvy může signalizovat potenciální problémy, které by mohly vyžadovat další monitoring nebo analýzu.

```
{
    "run_id": 1,
    "status": "completed",
    "summary": {
        "IP_address": "8.8.8.8",
        "pkts_send": 2,
        "pkts_received": 2,
        "pkts_lost": 0.0,
        "rtt_min": 38.753,
        "rtt_max": 577.212,
        "rtt_avg": 307.983,
        "rtt_stddev": 380.748,
        "jitter": 538.459
    },
    "details": [
        {
            "connected_time": "2024-12-01T13:52:23.104092Z",
            "response_time": "2024-12-01T13:52:23.681303Z",
            "rtt": 577.212,
            "bytes_received": 208,
            "status_msg": "Echo Reply",
            "status_code": 0
        },
        {
            "connected_time": "2024-12-01T13:52:23.681686Z",
            "response_time": "2024-12-01T13:52:23.720438Z",
            "rtt": 38.753,
            "bytes_received": 208,
            "status_msg": "Echo Reply",
            "status_code": 0
        }
    ]
}
```

Většina testů může selhat, což znamená, že testovaná služba je nedostupná, nechová se korektně, nebo jsou parametry testu zadány nesprávně. Příkladem testu, kdy je zadán nedostupný uzel je následující výpis. Výsledek testu sice ukazuje, že test byl úspěšně dokončen (status = "completed"), ale nedošlo k žádnému úspěšnému přijetí paketů. Testování bylo zaměřeno na hostitele s IP adresou 192.168.1.11.

```
{
    "run_id": 2,
    "status": "completed",
    "summary": {
        "IP_address": "192.168.1.11",
        "pkts_send": 2,
        "pkts_received": 0,
        "pkts_lost": 100.0,
        "rtt_min": 0,
        "rtt_max": 0,
        "rtt_avg": 0.0,
        "rtt_stddev": 0.0,
        "jitter": 0.0
    },
    "details": [
        {
            "connected_time": "2024-12-01T14:15:11.302364Z",
            "response_time": "2024-12-01T14:15:14.352015Z",
            "rtt": 0,
            "bytes_received": 236,
            "status_msg": "Destination host unreachable",
            "status_code": 3
        },
        {
            "connected_time": "2024-12-01T14:15:14.352273Z",
            "response_time": "2024-12-01T14:15:17.388644Z",
            "rtt": 0,
            "bytes_received": 236,
            "status_msg": "Destination host unreachable",
            "status_code": 3
        }
    ]
}
```

Výsledky tohoto testu ukazují, že cílový hostitel 192.168.1.11 nebyl dostupný během provádění testu, což představují zprávy o nedostupnosti cílového hostitele a nulové RTT hodnoty. Vzhledem k tomu, že obě sondy nahlásily stav "Destination host unreachable", je pravděpodobné, že adresu hostitele nelze dosáhnout z důvodu síťové chyby, chyby v konfiguraci, nebo proto, že hostitel není v provozu, případně neodpovídá na echo request zprávy. V takovém případě je doporučeno provést další diagnostiku sítě nebo zkontrolovat nastavení hostitelského zařízení.

Závěr

Tato zpráva popisuje dosažení výsledku monitorovacího agenta, který byl, v souladu s časovým harmonogramem projektu, dosažen. Výsledek obsahuje všechny předem definované vlastnosti a splňuje požadavky plynoucí ze zadávací dokumentace projektu. Výsledek je dostupný v podobě open-source kódu na adrese https://rysavy-ondrej.github.io/project-inventor/. Práce na integraci výsledku do produktu Flowmon začnou až po ukončení implementace zbylých výsledků (V2 a V3), které je plánováno na konec roku 2025.

Na dalších stránkách se nachází přiložená dokumentace REST API rozhraní agenta, prostřednictvím kterého je možné komunikovat s agentem. Jedná se o export Swagger dokumentace, která byla vytisknuta do PDF formátu.

Následně se na konci dokumentu nachází popis jednotlivých implementovaných testů. Popis je součástí programátorského dokumentace zdrojových kódů implementace ve formátu Markdown a pro tyto účely byl tento popis také vyexportován do PDF formátu.

INVENTOR - agent 1.0.5 OAS 3.1

/openapi.json

This is a documentation of the Agent tool developed under the INVENTOR project.

Authorize

Auth Operations related to authentication and authorization.

POST `/auth/token` Post Token

Parameters Try it out

No parameters

Request body required application/x-www-form-urlencoded

grant_type

username * required
`string`

password * required
`string`

scope
`string`

client_id

client_secret

Responses

Code	Description	Links
200	Successful Response Media type application/json Controls `Accept` header. **Example Value** Schema `{ "access_token": "string" }`	*No links*
422	Validation Error Media type application/json **Example Value** Schema	*No links*

```
{
  "access_token": "string"
}
```

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/auth/time` Returns the current time on the agent.

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Successful Response	*No links*

Code	Description	Links
	Media type application/json Controls `Accept` header. **Example Value** Schema	

```
{
  "time": 0
}
```

Tests Operations with tests, such as creating, updating, and reading.

GET `/test/all` Retrieve all the tests specified on the agent.

Parameters

Try it out

Name	Description
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response Media type	*No links*

Code	Description	Links

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "tests": [
    {
      "description": "description",
      "key_ro": "random_value_1",
      "key_rw": "random_value_2",
      "name": "ping.1",
      "recovery_attempt_limit": 3,
      "recovery_interval": 30,
      "scheduling_from": 1,
      "scheduling_interval": 60,
      "scheduling_until": 123456,
      "state": "disabled",
      "test_params": "{}",
      "timeout": 60,
      "version": 1
    }
  ]
}
```

422 Validation Error *No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/test/{id_test}` Retrieve information about the test.

Parameters Try it out

Name	Description

Name	Description
id_test * required integer *(path)*	id_test
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response Media type **application/json** Controls `Accept` header. **Example Value** Schema	*No links*

```
{
  "description": "description",
  "key_ro": "random_value_1",
  "key_rw": "random_value_2",
  "name": "ping.1",
  "recovery_attempt_limit": 3,
  "recovery_interval": 30,
  "scheduling_from": 1,
  "scheduling_interval": 60,
  "scheduling_until": 123456,
  "state": "disabled",
  "test_params": "{}",
  "timeout": 60,
  "version": 1
}
```

Code	Description	Links
422	Validation Error Media type	*No links*

Code	Description	Links

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

PATCH `/test/{id_test}` Updates the test configuration.

Parameters Try it out

Name	Description
id_test * required `integer` *(path)*	id_test
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Request body required application/json

Example Value Schema

```
{
  "description": "description",
  "recovery_attempt_limit": 3,
  "recovery_interval": 30,
  "scheduling_from": 1,
  "scheduling_interval": 60,
  "scheduling_until": 123456,
  "state": "disabled",
  "test_params": "{}",
  "timeout": 60
}
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "description": "description",
  "key_ro": "random_value_1",
  "key_rw": "random_value_2",
  "name": "ping.1",
  "recovery_attempt_limit": 3,
  "recovery_interval": 30,
  "scheduling_from": 1,
  "scheduling_interval": 60,
  "scheduling_until": 123456,
  "state": "disabled",
  "test_params": "{}",
  "timeout": 60,
  "version": 1
}
```

Code	Description	Links
422	Validation Error	*No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
```

Code	Description	Links
	}	

GET `/test/{id_test}/full` Retrieve information from all tables about the test.

Parameters Try it out

Name	Description
id_test * required `integer` *(path)*	id_test
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "test": {
    "description": "description",
    "key_ro": "random_value_1",
    "key_rw": "random_value_2",
```

Code	Description	Links

```
      "name": "ping.1",
      "recovery_attempt_limit": 3,
      "recovery_interval": 30,
      "scheduling_from": 1,
      "scheduling_interval": 60,
      "scheduling_until": 123456,
      "state": "disabled",
      "test_params": "{}",
      "timeout": 60,
      "version": 1
    },
    "requests": [
      {
        "fk_tests": 0,
        "reason": "new",
        "recovery_attempt": 0,
        "added_time": 0,
        "id_request": 0
      }
    ],
    "events": [
```

422 Validation Error *No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/test/{id_test}/results` Returns the results for the specified test.

Parameters Try it out

Name	Description
id_test * required `integer` *(path)*	id_test
since_id * required `integer` *(query)*	since_id

Name	Description
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "results": [
    {
      "fk_tests": 0,
      "version": 0,
      "planned": 0,
      "started": 0,
      "finished": 0,
      "status": "success",
      "recovery_attempt": 0,
      "data": "string",
      "id_result": 0
    }
  ]
}
```

Code	Description	Links
422	Validation Error	*No links*

Media type

application/json

Example Value Schema

Code	Description	Links
	`{ "detail": [ { "loc": [ "string", 0 ], "msg": "string", "type": "string" } ] }`	

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/test/{id_test}/events` Returns all the planned events for the specified test.

Parameters

Try it out

Name	Description
id_test * required integer *(path)*	id_test
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response Media type	*No links*

Code	Description	Links

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "events": [
    {
      "run_at": 0,
      "source": "string",
      "recovery_attempt": 0,
      "id_event": 0,
      "fk_tests": 0
    }
  ]
}
```

422 Validation Error *No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

POST `/test/{id_test}/request` Create a new test request for an event in the calendar.

Parameters Try it out

Name	Description
id_test * required `integer` *(path)*	id_test
authorization-time * required `string` *(header)*	authorization-time

Name	Description
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response Media type **application/json** Controls `Accept` header. **Example Value** Schema `0`	*No links*
422	Validation Error Media type **application/json** **Example Value** Schema	*No links*

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/test/{id_test}/old_params` Returns the previous test configuration for a given test and version.

Parameters

Try it out

Name	Description
id_test * required `integer` *(path)*	id_test
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "old_params": [
    {
      "fk_tests": 0,
      "version": 0,
      "changed": 0,
      "test_params": "string",
      "id_old_params": 0
    }
  ]
}
```

Code	Description	Links
422	Validation Error	*No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/test/{id_test}/old_params/{version}` Returns the all known previous test configurations for a given test.

Parameters

Try it out

Name	Description
id_test * required `integer` *(path)*	id_test
version * required `integer` *(path)*	version
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "fk_tests": 0,
  "version": 0,
  "changed": 0,
  "test_params": "string",
  "id_old_params": 0
}
```

Code	Description	Links
422	Validation Error	*No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

POST `/test` Create a new test.

Parameters

Try it out

Name	Description
authorization-time * required `string` *(header)*	authorization-time

Name	Description
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Request body required   application/json

Example Value Schema

```
{
  "description": "description",
  "key_ro": "random_value_1",
  "key_rw": "random_value_2",
  "name": "ping.1",
  "recovery_attempt_limit": 3,
  "recovery_interval": 30,
  "scheduling_from": 1,
  "scheduling_interval": 60,
  "scheduling_until": 123456,
  "state": "disabled",
  "test_params": "{}",
  "timeout": 60,
  "version": 1
}
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "description": "description",
  "key_ro": "random_value_1",
  "key_rw": "random_value_2",
  "name": "ping.1",
  "recovery_attempt_limit": 3,
  "recovery_interval": 30,
```

Code	Description	Links
	(continued)	

```
    "scheduling_from": 1,
    "scheduling_interval": 60,
    "scheduling_until": 123456,
    "state": "disabled",
    "test_params": "{}",
    "timeout": 60,
    "version": 1
  }
```

422 Validation Error — *No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Multiple Results

Operations that work with results from multiple tests at once.

POST `/multi-results/init` Create a new MultiResult record.

Parameters Try it out

Name	Description
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac

Name	Description
authorization-nonce * required string *(header)*	authorization-nonce

Request body required — application/json

Example Value Schema

```
{
  "key": "string"
}
```

Responses

Code	Description	Links
200	Successful Response	*No links*
422	Validation Error	*No links*

200 Successful Response

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "id_multi_result": 0
}
```

422 Validation Error

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
```

Code	Description	Links
	] }	

POST `/multi-results/{multi_results_id}` Add a new test to the multi result record.

Parameters

Try it out

Name	Description
multi_results_id * required integer *(path)*	multi_results_id
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Request body required

application/json

Example Value Schema

```
{
  "fk_tests": 0,
  "hash": "string"
}
```

Responses

Code	Description	Links

Code	Description	Links
200	Successful Response	*No links*
422	Validation Error	*No links*

200 Successful Response

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "test_ids": "string"
}
```

422 Validation Error

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/multi-results/{multi_results_id}` Retrieve the results related to the specified multi results record.

Parameters Try it out

Name	Description
multi_results_id * required `integer` *(path)*	multi_results_id
since_id * required `integer` *(query)*	since_id

Name	Description
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "results": {
    "additionalProp1": {
      "results": [
        {
          "fk_tests": 0,
          "version": 0,
          "planned": 0,
          "started": 0,
          "finished": 0,
          "status": "success",
          "recovery_attempt": 0,
          "data": "string",
          "id_result": 0
        }
      ]
    },
    "additionalProp2": {
      "results": [
        {
          "fk_tests": 0,
          "version": 0,
          "planned": 0,
          "started": 0,
          "finished": 0,
```

Code	Description	Links

```
        "status": "success",
        "recovery_attempt": 0,
```

Code	Description	Links
422	Validation Error	*No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

System

Operation that are not related to specific test, but are system-wide.

GET `/system/config` Returns system information about the agent.

Parameters

Try it out

Name	Description
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*
422	Validation Error	*No links*

200 Successful Response

Media type: **application/json**

Controls `Accept` header.

Example Value Schema

```
{
  "options": {
    "additionalProp1": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "additionalProp2": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "additionalProp3": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    }
  }
}
```

422 Validation Error

Media type: **application/json**

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

PATCH `/system/config` Updates the values of the specified config options.

Parameters Try it out

Name	Description
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Request body required

application/json

Example Value Schema

```
{
  "additionalProp1": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "additionalProp2": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  },
  "additionalProp3": {
    "additionalProp1": "string",
    "additionalProp2": "string",
    "additionalProp3": "string"
  }
}
```

Responses

Code	Description	Links
200	Successful Response Media type application/json Controls `Accept` header. **Example Value** Schema	*No links*

Code	Description	Links

```
{
  "options": {
    "additionalProp1": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "additionalProp2": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "additionalProp3": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    }
  }
}
```

422 Validation Error *No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/system/config/all` Returns all configuration options and their values.

Parameters Try it out

Name	Description
authorization-time * required `string` *(header)*	authorization-time

Name	Description
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "options": {
    "additionalProp1": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "additionalProp2": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "additionalProp3": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    }
  }
}
```

Code	Description	Links
422	Validation Error	*No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
```

Code **Description** **Links**

```
      {
        "loc": [
          "string",
          0
        ],
        "msg": "string",
        "type": "string"
      }
    ]
  }
```

GET `/system/orchestrators` Returns all the orchestrators that ever connected with the agent.

Parameters

Try it out

Name	Description
authorization-time * required string *(header)*	authorization-time
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response Media type **application/json** Controls `Accept` header. **Example Value** Schema `{`	*No links*

Code	Description	Links

```
  "orchestrators": [
    {
      "name": "string",
      "last_seen": 0,
      "id_orchestrator": 0
    }
  ]
}
```

422 Validation Error *No links*

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/system/logs` Returns the log records (all type of messages) since the specified datetime.

Parameters Try it out

Name	Description
since * required `string` *(query)*	since
max_size *(query)*	max_size
compression_alg *(query)*	compression_alg
authorization-time * required `string` *(header)*	authorization-time

Name	Description
authorization-hmac * required string *(header)*	authorization-hmac
authorization-nonce * required string *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response Media type **application/json** Controls `Accept` header. **Example Value** Schema	*No links*

```
{
  "data": "string",
  "compression_alg": "zlib_base85",
  "last_datetime": "string",
  "more_data": true
}
```

Code	Description	Links
422	Validation Error Media type **application/json** **Example Value** Schema	*No links*

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/system/logs/stats` Returns the statistics about the log records calculated from the last N minutes.

Parameters

Try it out

Name	Description
minutes * required `integer` *(query)*	minutes
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string` *(header)*	authorization-nonce

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "debug": 0,
  "info": 0,
  "warning": 0,
  "error": 0,
  "critical": 0,
  "unknown": 0
}
```

Code	Description	Links
422	Validation Error Media type application/json **Example Value** Schema	*No links*

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

GET `/system/accounting` Returns the accounting records since the specified datetime.

Parameters

Try it out

Name	Description
since * required `string` *(query)*	since
max_size *(query)*	max_size
compression_alg *(query)*	compression_alg
authorization-time * required `string` *(header)*	authorization-time
authorization-hmac * required `string` *(header)*	authorization-hmac
authorization-nonce * required `string`	authorization-nonce

Name	Description
(header)	

Responses

Code	Description	Links
200	Successful Response	*No links*
422	Validation Error	*No links*

200 Successful Response

Media type

application/json

Controls `Accept` header.

Example Value Schema

```
{
  "data": "string",
  "compression_alg": "zlib_base85",
  "last_datetime": "string",
  "more_data": true
}
```

422 Validation Error

Media type

application/json

Example Value Schema

```
{
  "detail": [
    {
      "loc": [
        "string",
        0
      ],
      "msg": "string",
      "type": "string"
    }
  ]
}
```

Schemas

Accounting ^ Collapse all `object`

`data`* ^ Collapse all `string`

Records from the log file (can be compressed).

compression_alg Collapse all (string | null)

Name of the compression algorithm used to compress the data.

Any of Collapse all (string | null)

#0 CompressionAlg Expand all string

#1 null

last_datetime* Collapse all (string | null)

Datetime of the last record. This value is used to download more records in the future.

Any of Collapse all (string | null)

#0 string

#1 null

more_data* Collapse all boolean

If there is too much data, not all of them are downloaded at once. This attribute indicated, whether there are more data that could be sent in this request.

Body_post_token_auth_token_post Collapse all object

grant_type Collapse all (string | null)

Any of Collapse all (string | null)

#0 string matches password

#1 null

username* string

password* string

scope Collapse all string

Default ""

client_id Collapse all (string | null)

Any of Collapse all (string | null)

#0 string

#1 null

client_secret Collapse all (string | null)

Any of Collapse all (string | null)

#0 string

#1 null

CompressionAlg Collapse all string

Const "zlib_base85"

Config Collapse all object

options* Collapse all object

Dictionary with sections as keys and values as nested dictionaries with various config options and their values.

`Additional properties` ^ Collapse all `object`

`Additional properties` `string`

ConfigChanges ^ Collapse all `object`

`options*` ^ Collapse all `object`

Dictionary with sections as keys and values as nested dictionaries with information whether the options was changed or added.

`Additional properties` ^ Collapse all `object`

`Additional properties` `string`

Event ^ Collapse all `object`

`run_at*` ^ Collapse all `number`

Time when the event should be executed.

`source*` ^ Collapse all `string`

What's the source of the event - what cased that the event was planned.

`recovery_attempt*` ^ Collapse all `integer`

How many times the test failed before this test.

`id_event*` ^ Collapse all `integer`

ID of the event.

`fk_tests*` ^ Collapse all `integer`

ID of the test.

Events ^ Collapse all `object`

`events*` ^ Collapse all `array<object>`

List of planned events.

`Items` ^ Collapse all `object`

`run_at*` ^ Collapse all `number`

Time when the event should be executed.

`source*` ^ Collapse all `string`

What's the source of the event - what cased that the event was planned.

`recovery_attempt*` ^ Collapse all `integer`

How many times the test failed before this test.

`id_event*` ^ Collapse all `integer`

ID of the event.

`fk_tests*` ^ Collapse all `integer`

ID of the test.

HTTPValidationError ^ Collapse all `object`

`detail` ^ Collapse all `array<object>`

`Items` ^ Collapse all `object`

loc* ^ Collapse all array<(string | integer)>

Items ^ Collapse all (string | integer)

Any of ^ Collapse all (string | integer)

#0 string

#1 integer

msg* string

type* string

Logs ^ Collapse all object

data* ^ Collapse all string

Records from the log file (can be compressed).

compression_alg ^ Collapse all (string | null)

Name of the compression algorithm used to compress the data.

Any of ^ Collapse all (string | null)

#0 CompressionAlg > Expand all string

#1 null

last_datetime* ^ Collapse all (string | null)

Datetime of the last record. This value is used to download more records in the future.

Any of ^ Collapse all (string | null)

#0 string

#1 null

more_data* ^ Collapse all boolean

If there is too much data, not all of them are downloaded at once. This attribute indicated, whether there are more data that that could be sent in this request.

LogsStats ^ Collapse all object

debug* ^ Collapse all integer

Amount of rows with DEBUG severity.

info* ^ Collapse all integer

Amount of rows with INFO severity.

warning* ^ Collapse all integer

Amount of rows with WARNING severity.

error* ^ Collapse all integer

Amount of rows with ERROR severity.

critical* ^ Collapse all integer

Amount of rows with CRITICAL severity.

unknown* ^ Collapse all integer

Amount of rows with unknown severity. This should be always 0.

MultiResult ^ Collapse all object

results* ^ Collapse all `object`

Test results for the specified test.

Additional properties ^ Collapse all `object`

results* ^ Collapse all `array<object>`

Test results for the specified test.

Items ^ Collapse all `object`

fk_tests* ^ Collapse all `integer`

ID of the test.

version* ^ Collapse all `integer`

Version of the test, that was used to run the test.

planned* ^ Collapse all `number`

Time when the test was planned to start.

started* ^ Collapse all `number`

Time when the test actually started.

finished* ^ Collapse all `number`

Time when the execution of the test finished.

status* ^ Collapse all `string`

Specifies with what status the test has finished.

Allowed values
"success" "terminated" "error" "crashed"

recovery_attempt* ^ Collapse all `integer`

How many times the test failed before this test.

data ^ Collapse all `(string | null)`

Contains all the result data from the test.

Any of ^ Collapse all `(string | null)`

#0 `string`

#1 `null`

id_result* ^ Collapse all `integer`

ID of the result.

last_checked_id* ^ Collapse all `integer`

ID of the test.

MultiResultAddTestInput ^ Collapse all `object`

fk_tests* ^ Collapse all `integer`

ID of the test.

hash* ^ Collapse all `string`

Hash calculated from the multi results key.

MultiResultCreate ^ Collapse all `object`

key* ^ Collapse all `string`

Authorization key used to download results from multiple tests at once.

MultiResultId ^ Collapse all `object`

`id_multi_result`* ^ Collapse all `integer`

ID of the multi result.

MultiResultTestsIds ^ Collapse all `object`

`test_ids`* ^ Collapse all `string`

List of test IDs related to the multi result.

OldParams ^ Collapse all `object`

`fk_tests`* ^ Collapse all `integer`

ID of the test.

`version`* ^ Collapse all `integer`

Version of the test.

`changed`* ^ Collapse all `number`

Timestamp when the test was changed.

`test_params`* ^ Collapse all `string`

Parameters of the test for the specified version.

`id_old_params`* ^ Collapse all `integer`

ID of the old parameter record.

OldParamsList ^ Collapse all `object`

`old_params`* ^ Collapse all `array<object>`

List of old test parameters.

`Items` ^ Collapse all `object`

`fk_tests`* ^ Collapse all `integer`

ID of the test.

`version`* ^ Collapse all `integer`

Version of the test.

`changed`* ^ Collapse all `number`

Timestamp when the test was changed.

`test_params`* ^ Collapse all `string`

Parameters of the test for the specified version.

`id_old_params`* ^ Collapse all `integer`

ID of the old parameter record.

Orchestrator ^ Collapse all `object`

`name`* ^ Collapse all `string`

Human readable orchestrator name.

`last_seen`* ^ Collapse all `number`

The last time when the orchestrator has either authenticated or used any endpoint.

`id_orchestrator`* ^ Collapse all `integer`

ID of the orchestrator.

Orchestrators ^ Collapse all `object`

`orchestrators`* ^ Collapse all `array<object>`

`Items` ^ Collapse all `object`

`name`* ^ Collapse all `string`

Human readable orchestrator name.

`last_seen`* ^ Collapse all `number`

The last time when the orchestrator has either authenticated or used any endpoint.

`id_orchestrator`* ^ Collapse all `integer`

ID of the orchestrator.

Request ^ Collapse all `object`

`fk_tests`* ^ Collapse all `integer`

ID of the test.

`reason`* ^ Collapse all `string`

Specifies why the request was created.

Allowed values `"new"` `"update"` `"failed"`

`recovery_attempt`* ^ Collapse all `integer`

Specifies the recovery counter, which recovery test it is.

`added_time`* ^ Collapse all `number`

Time when the request has been added.

`id_request`* ^ Collapse all `integer`

ID of the request.

RequestReason ^ Collapse all `string`

Allowed values `"new"` `"update"` `"failed"`

Result ^ Collapse all `object`

`fk_tests`* ^ Collapse all `integer`

ID of the test.

`version`* ^ Collapse all `integer`

Version of the test, that was used to run the test.

`planned`* ^ Collapse all `number`

Time when the test was planned to start.

`started`* ^ Collapse all `number`

Time when the test actually started.

`finished`* ^ Collapse all `number`

Time when the execution of the test finished.

`status`* ^ Collapse all `string`

Specifies with what status the test has finished.

Allowed values `"success"` `"terminated"` `"error"` `"crashed"`

`recovery_attempt`* ^ Collapse all `integer`

How many times the test failed before this test.

`data` ^ Collapse all `(string | null)`

Contains all the result data from the test.

Any of ^ Collapse all `(string | null)`

`#0` `string`

`#1` `null`

`id_result`* ^ Collapse all `integer`

ID of the result.

ResultStatus ^ Collapse all `string`

Allowed values `"success"` `"terminated"` `"error"` `"crashed"`

Results ^ Collapse all `object`

`results`* ^ Collapse all `array<object>`

Test results for the specified test.

`Items` ^ Collapse all `object`

`fk_tests`* ^ Collapse all `integer`

ID of the test.

`version`* ^ Collapse all `integer`

Version of the test, that was used to run the test.

`planned`* ^ Collapse all `number`

Time when the test was planned to start.

`started`* ^ Collapse all `number`

Time when the test actually started.

`finished`* ^ Collapse all `number`

Time when the execution of the test finished.

`status`* ^ Collapse all `string`

Specifies with what status the test has finished.

Allowed values `"success"` `"terminated"` `"error"` `"crashed"`

`recovery_attempt`* ^ Collapse all `integer`

How many times the test failed before this test.

`data` ^ Collapse all `(string | null)`

Contains all the result data from the test.

Any of ^ Collapse all `(string | null)`

`#0` `string`

`#1` `null`

`id_result`* ^ Collapse all `integer`

ID of the result.

Run ^ Collapse all object

fk_tests* ^ Collapse all integer

ID of the test.

version ^ Collapse all (integer | null)

Version of the test which is running.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

state* ^ Collapse all string

Specifies the state of the run.

Allowed values "waiting" "running" "terminating" "killing" "zombie"

pid ^ Collapse all (integer | null)

PID of the new created process that runs the test.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

planned* ^ Collapse all number

Time when the instruction to run the test was created.

started ^ Collapse all (number | null)

Time when the test actually started.

Any of ^ Collapse all (number | null)

#0 number

#1 null

deadline ^ Collapse all (number | null)

Time until which the test must end, later terminated or killed.

Any of ^ Collapse all (number | null)

#0 number

#1 null

recovery_attempt* ^ Collapse all integer

How many times the test failed before this test.

id_run* ^ Collapse all integer

ID of the run.

RunState ^ Collapse all string

Allowed values "waiting" "running" "terminating" "killing" "zombie"

Test ^ Collapse all object

description* ^ Collapse all string

Description of the test defined by the user.

state* ^ Collapse all string

State of the test.

Allowed values
"enabled" "disabled" "deleted" "migrating_from" "migrating_to"

test_params* ^ Collapse all string

Parameters of the test that specifies how the test should be executed.

timeout* ^ Collapse all integer

Specifies in what time the test must finish, otherwise it will be terminated.

scheduling_interval* ^ Collapse all (integer | null)

Interval in which the test should be scheduled, or None if the test is only one-time run.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

scheduling_from* ^ Collapse all (number | null)

Time from when the test should be scheduled to execute.

Any of ^ Collapse all (number | null)

#0 number

#1 null

scheduling_until* ^ Collapse all (number | null)

Time until when the test should be scheduled to execute.

Any of ^ Collapse all (number | null)

#0 number

#1 null

recovery_interval* ^ Collapse all integer

Interval in which the recovery test should be scheduled.

recovery_attempt_limit* ^ Collapse all (integer | null)

How many time the recovery test should be executed. Zero means no recovery test, None means infinite tests.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

name* ^ Collapse all string

Name of the test.

version* ^ Collapse all integer

The actual version of the test.

key_ro* ^ Collapse all string

Authorization key used to identify the orchestrator that can read the data about the test.

key_rw* ^ Collapse all string

Authorization key used to identify the orchestrator that can change the test parameters.

id_test* ^ Collapse all integer

ID of the test.

last_started_time ^ Collapse all (number | null)

Time when the last test was executed.

Any of Collapse all `(number | null)`

`#0 number`

`#1 null`

`last_result_time` Collapse all `(number | null)`

Time when the last execution of the test was finished.

Any of Collapse all `(number | null)`

`#0 number`

`#1 null`

`last_result_status*` Collapse all `(string | null)`

Result from the last test execution.

Any of Collapse all `(string | null)`

`#0 ResultStatus` Expand all `string`

`#1 null`

`last_downloaded_time` Collapse all `(number | null)`

Time when the results for the test were last downloaded. This includes request when there are no results available.

Any of Collapse all `(number | null)`

`#0 number`

`#1 null`

TestCreate Collapse all `object`

`description*` Collapse all `string`

Description of the test defined by the user.

`state*` Collapse all `string`

State of the test.

Allowed values
`"enabled"` `"disabled"` `"deleted"` `"migrating_from"` `"migrating_to"`

`test_params*` Collapse all `string`

Parameters of the test that specifies how the test should be executed.

`timeout*` Collapse all `integer`

Specifies in what time the test must finish, otherwise it will be terminated.

`scheduling_interval*` Collapse all `(integer | null)`

Interval in which the test should be scheduled, or None if the test is only one-time run.

Any of Collapse all `(integer | null)`

`#0 integer`

`#1 null`

`scheduling_from*` Collapse all `(number | null)`

Time from when the test should be scheduled to execute.

Any of Collapse all `(number | null)`

`#0 number`

`#1 null`

scheduling_until* ^ Collapse all `(number | null)`

Time until when the test should be scheduled to execute.

Any of ^ Collapse all `(number | null)`

#0 `number`

#1 `null`

recovery_interval* ^ Collapse all `integer`

Interval in which the recovery test should be scheduled.

recovery_attempt_limit* ^ Collapse all `(integer | null)`

How many time the recovery test should be executed. Zero means no recovery test, None means infinite tests.

Any of ^ Collapse all `(integer | null)`

#0 `integer`

#1 `null`

name* ^ Collapse all `string`

Name of the test.

version* ^ Collapse all `integer`

The actual version of the test.

key_ro* ^ Collapse all `string`

Authorization key used to identify the orchestrator that can read the data about the test.

key_rw* ^ Collapse all `string`

Authorization key used to identify the orchestrator that can change the test parameters.

TestFullInfo ^ Collapse all `object`

test* ^ Collapse all `object`

Test record.

description* ^ Collapse all `string`

Description of the test defined by the user.

state* ^ Collapse all `string`

State of the test.

Allowed values
`"enabled"` `"disabled"` `"deleted"` `"migrating_from"` `"migrating_to"`

test_params* ^ Collapse all `string`

Parameters of the test that specifies how the test should be executed.

timeout* ^ Collapse all `integer`

Specifies in what time the test must finish, otherwise it will be terminated.

scheduling_interval* ^ Collapse all `(integer | null)`

Interval in which the test should be scheduled, or None if the test is only one-time run.

Any of ^ Collapse all `(integer | null)`

#0 `integer`

#1 `null`

scheduling_from* ^ Collapse all `(number | null)`

Time from when the test should be scheduled to execute.

Any of ^ Collapse all `(number | null)`

#0 number

#1 null

scheduling_until* ^ Collapse all (number | null)

Time until when the test should be scheduled to execute.

Any of ^ Collapse all (number | null)

#0 number

#1 null

recovery_interval* ^ Collapse all integer

Interval in which the recovery test should be scheduled.

recovery_attempt_limit* ^ Collapse all (integer | null)

How many time the recovery test should be executed. Zero means no recovery test, None means infinite tests.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

name* ^ Collapse all string

Name of the test.

version* ^ Collapse all integer

The actual version of the test.

key_ro* ^ Collapse all string

Authorization key used to identify the orchestrator that can read the data about the test.

key_rw* ^ Collapse all string

Authorization key used to identify the orchestrator that can change the test parameters.

id_test* ^ Collapse all integer

ID of the test.

last_started_time ^ Collapse all (number | null)

Time when the last test was executed.

Any of ^ Collapse all (number | null)

#0 number

#1 null

last_result_time ^ Collapse all (number | null)

Time when the last execution of the test was finished.

Any of ^ Collapse all (number | null)

#0 number

#1 null

last_result_status* ^ Collapse all (string | null)

Result from the last test execution.

Any of ^ Collapse all (string | null)

#0 ResultStatus > Expand all string

#1 null

last_downloaded_time ^ Collapse all (number | null)

Time when the results for the test were last downloaded. This includes request when there are no results available.

Any of ^ Collapse all `(number | null)`

`#0` `number`

`#1` `null`

`requests*` ^ Collapse all `array<object>`

Request records for the specified test.

`Items` ^ Collapse all `object`

`fk_tests*` ^ Collapse all `integer`

ID of the test.

`reason*` ^ Collapse all `string`

Specifies why the request was created.

Allowed values `"new"` `"update"` `"failed"`

`recovery_attempt*` ^ Collapse all `integer`

Specifies the recovery counter, which recovery test it is.

`added_time*` ^ Collapse all `number`

Time when the request has been added.

`id_request*` ^ Collapse all `integer`

ID of the request.

`events*` ^ Collapse all `array<object>`

Event records for the specified test.

`Items` ^ Collapse all `object`

`run_at*` ^ Collapse all `number`

Time when the event should be executed.

`source*` ^ Collapse all `string`

What's the source of the event - what cased that the event was planned.

`recovery_attempt*` ^ Collapse all `integer`

How many times the test failed before this test.

`id_event*` ^ Collapse all `integer`

ID of the event.

`fk_tests*` ^ Collapse all `integer`

ID of the test.

`runs*` ^ Collapse all `array<object>`

Run records for the specified test.

`Items` ^ Collapse all `object`

`fk_tests*` ^ Collapse all `integer`

ID of the test.

`version` ^ Collapse all `(integer | null)`

Version of the test which is running.

Any of ^ Collapse all `(integer | null)`

`#0` `integer`

`#1` `null`

`state*` ^ Collapse all `string`

Specifies the state of the run.

Allowed values

"waiting" "running" "terminating" "killing" "zombie"

pid ^ Collapse all (integer | null)

PID of the new created process that runs the test.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

planned* ^ Collapse all number

Time when the instruction to run the test was created.

started ^ Collapse all (number | null)

Time when the test actually started.

Any of ^ Collapse all (number | null)

#0 number

#1 null

deadline ^ Collapse all (number | null)

Time until which the test must end, later terminated or killed.

Any of ^ Collapse all (number | null)

#0 number

#1 null

recovery_attempt* ^ Collapse all integer

How many times the test failed before this test.

id_run* ^ Collapse all integer

ID of the run.

old_params* ^ Collapse all array<object>

Old parameters records for the specified test.

Items ^ Collapse all object

fk_tests* ^ Collapse all integer

ID of the test.

version* ^ Collapse all integer

Version of the test.

changed* ^ Collapse all number

Timestamp when the test was changed.

test_params* ^ Collapse all string

Parameters of the test for the specified version.

id_old_params* ^ Collapse all integer

ID of the old parameter record.

results* ^ Collapse all array<object>

Result records for the specified test.

Items ^ Collapse all object

fk_tests* ^ Collapse all integer

ID of the test.

version* ^ Collapse all integer

Version of the test, that was used to run the test.

planned* ^ Collapse all number

Time when the test was planned to start.

started* ^ Collapse all `number`

Time when the test actually started.

finished* ^ Collapse all `number`

Time when the execution of the test finished.

status* ^ Collapse all `string`

Specifies with what status the test has finished.

Allowed values "success" "terminated" "error" "crashed"

recovery_attempt* ^ Collapse all `integer`

How many times the test failed before this test.

data ^ Collapse all `(string | null)`

Contains all the result data from the test.

Any of ^ Collapse all `(string | null)`

#0 `string`

#1 `null`

id_result* ^ Collapse all `integer`

ID of the result.

TestState ^ Collapse all `string`

Allowed values
"enabled" "disabled" "deleted" "migrating_from" "migrating_to"

TestUpdate ^ Collapse all `object`

description* ^ Collapse all `string`

Description of the test defined by the user.

state* ^ Collapse all `string`

State of the test.

Allowed values
"enabled" "disabled" "deleted" "migrating_from" "migrating_to"

test_params* ^ Collapse all `string`

Parameters of the test that specifies how the test should be executed.

timeout* ^ Collapse all `integer`

Specifies in what time the test must finish, otherwise it will be terminated.

scheduling_interval* ^ Collapse all `(integer | null)`

Interval in which the test should be scheduled, or None if the test is only one-time run.

Any of ^ Collapse all `(integer | null)`

#0 `integer`

#1 `null`

scheduling_from* ^ Collapse all `(number | null)`

Time from when the test should be scheduled to execute.

Any of ^ Collapse all `(number | null)`

#0 `number`

#1 null

scheduling_until* ^ Collapse all (number | null)

Time until when the test should be scheduled to execute.

Any of ^ Collapse all (number | null)

#0 number

#1 null

recovery_interval* ^ Collapse all integer

Interval in which the recovery test should be scheduled.

recovery_attempt_limit* ^ Collapse all (integer | null)

How many time the recovery test should be executed. Zero means no recovery test, None means infinite tests.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

Tests ^ Collapse all object

tests* ^ Collapse all array<object>

List of tests.

Items ^ Collapse all object

description* ^ Collapse all string

Description of the test defined by the user.

state* ^ Collapse all string

State of the test.

Allowed values
"enabled" "disabled" "deleted" "migrating_from" "migrating_to"

test_params* ^ Collapse all string

Parameters of the test that specifies how the test should be executed.

timeout* ^ Collapse all integer

Specifies in what time the test must finish, otherwise it will be terminated.

scheduling_interval* ^ Collapse all (integer | null)

Interval in which the test should be scheduled, or None if the test is only one-time run.

Any of ^ Collapse all (integer | null)

#0 integer

#1 null

scheduling_from* ^ Collapse all (number | null)

Time from when the test should be scheduled to execute.

Any of ^ Collapse all (number | null)

#0 number

#1 null

scheduling_until* ^ Collapse all (number | null)

Time until when the test should be scheduled to execute.

Any of ^ Collapse all `(number | null)`

`#0` `number`

`#1` `null`

`recovery_interval`* ^ Collapse all `integer`

Interval in which the recovery test should be scheduled.

`recovery_attempt_limit`* ^ Collapse all `(integer | null)`

How many time the recovery test should be executed. Zero means no recovery test, None means infinite tests.

Any of ^ Collapse all `(integer | null)`

`#0` `integer`

`#1` `null`

`name`* ^ Collapse all `string`

Name of the test.

`version`* ^ Collapse all `integer`

The actual version of the test.

`key_ro`* ^ Collapse all `string`

Authorization key used to identify the orchestrator that can read the data about the test.

`key_rw`* ^ Collapse all `string`

Authorization key used to identify the orchestrator that can change the test parameters.

`id_test`* ^ Collapse all `integer`

ID of the test.

`last_started_time` ^ Collapse all `(number | null)`

Time when the last test was executed.

Any of ^ Collapse all `(number | null)`

`#0` `number`

`#1` `null`

`last_result_time` ^ Collapse all `(number | null)`

Time when the last execution of the test was finished.

Any of ^ Collapse all `(number | null)`

`#0` `number`

`#1` `null`

`last_result_status`* ^ Collapse all `(string | null)`

Result from the last test execution.

Any of ^ Collapse all `(string | null)`

`#0` **`ResultStatus`** > Expand all `string`

`#1` `null`

`last_downloaded_time` ^ Collapse all `(number | null)`

Time when the results for the test were last downloaded. This includes request when there are no results available.

Any of ^ Collapse all `(number | null)`

`#0` `number`

`#1` `null`

Time ^ Collapse all `object`

- `time`* ^ Collapse all `number`
  - Current time on the agent.

Token ^ Collapse all `object`

- `access_token`* ^ Collapse all `string`
  - Encrypted token data.

ValidationError ^ Collapse all `object`

- `loc`* ^ Collapse all `array<(string | integer)>`
  - `Items` ^ Collapse all `(string | integer)`
    - Any of ^ Collapse all `(string | integer)`
      - `#0` `string`
      - `#1` `integer`
- `msg`* `string`
- `type`* `string`

Network Ping Test

A PING test is a basic network diagnostic tool used to verify the reachability of a host on an IP network, measuring the round-trip time (RTT) for messages sent from the originating host to a destination computer. The test operates by sending Internet Control Message Protocol (ICMP) echo request packets to the target host and waiting for it to send back an echo reply. This test is commonly used to check network connectivity and performance.

Requirements

Library	Version
icmplib	3.0.4

The test needs to be run with root privileges (e.g., using `sudo`) or with elevated privileges, because it needs to be able to send ICMP messages and have access to raw sockets. The ping utility on linux doesn't require `sudo` because it has necessary setuid permission, allowing regular users to send ICMP packets without elevated privileges.

Input

```
message PingTestConfig {
  string target_host = 1; // The IP address or hostname of the target.
  int32 packet_size = 2; // The size of the ICMP payload in bytes.
  int32 packet_count = 3; // The number of ICMP packets to send.
  float interpacket_delay = 4; // The delay between probes in seconds.
  float timeout = 5; // The maximum time to wait for a response in seconds.
}
```

Output

```
// Top-level message representing the entire test result
message PingTestResult {
  int32 run_id = 1;             // The identification of the test instance.
  TestStatus status = 2; // The status of the test (e.g., "completed")
  PingTestSummary summary = 2; // Nested message for summary details
  repeated PingTestDetails details = 3; // Array of Detail messages for each prob
}

// Summary of the test results
message PingTestSummary {
  int32 pkts_sent = 1; // Total packets sent
  int32 pkts_received = 2; // Total packets received
  float pkts_lost = 3; // Percentage of packets lost
  float rtt_min = 4; // Minimum round trip time in milliseconds
  float rtt_max = 5; // Maximum round trip time in milliseconds
  float rtt_avg = 6; // Average round trip time in milliseconds
  float rtt_stdev = 7; // Standard deviation of round trip times in milliseconds
  float jitter = 8; // Jitter in milliseconds
}

// Details of each individual probe
message PingTestDetails {
  string IP_address = 1; // The IP address of the target host
  int32 connected_time = 2; // Timestamp of the connection
  int32 response_time = 3; // Timestamp of the response
  float rtt = 4; // Round Trip Time in milliseconds
  float bytes_received = 5; // Number of bytes received
  string status_msg = 6; // Response code or message (e.g., "echo_reply")
  int32 status_code = 7; // ICMP code
}
```

Response code to string mapping:

Code	Message
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect Message
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded
12	Parameter Problem
13	Timestamp Request
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
42	Extended Echo Request
43	Extended Echo Reply

How to run simple test

```
sudo pytest monitor_ping.py
```

The result will be in the `test` directory in file `output.json` For running the test you need `sudo` permissions just as you would need them for running the script itself.

Examples

INPUT:

```
{
    "target_host": "8.8.8.8",
    "packet_size": 200,
    "packet_count": 2,
    "interpacket_delay": 3,
    "timeout": 5
}
```

OUTPUT:
```
{
    "run_id": 1,
    "status": "completed",
    "summary": {
        "IP_address": "8.8.8.8",
        "pkts_send": 2,
        "pkts_received": 2,
        "pkts_lost": 0.0,
        "rtt_min": 38.753,
        "rtt_max": 577.212,
        "rtt_avg": 307.983,
        "rtt_stddev": 380.748,
        "jitter": 538.459
    },
    "details": [
        {
            "connected_time": "2024-12-01T13:52:23.104092Z",
            "response_time": "2024-12-01T13:52:23.681303Z",
            "rtt": 577.212,
            "bytes_received": 208,
            "status_msg": "Echo Reply",
            "status_code": 0
        },
        {
            "connected_time": "2024-12-01T13:52:23.681686Z",
            "response_time": "2024-12-01T13:52:23.720438Z",
            "rtt": 38.753,
            "bytes_received": 208,
            "status_msg": "Echo Reply",
            "status_code": 0
        }
    ]
}
```
Example (failed test due to host unreachable)

INPUT:
```
{
    "target_host": "192.168.1.11",
    "packet_size": 200,
    "packet_count": 2,
    "interpacket_delay": 3,
    "timeout": 5
}
```
OUTPUT:
```
{
    "run_id": 1,
    "status": "completed",
    "summary": {
        "IP_address": "192.168.1.11",
        "pkts_send": 2,
        "pkts_received": 0,
        "pkts_lost": 100.0,
        "rtt_min": 0,
        "rtt_max": 0,
        "rtt_avg": 0.0,
        "rtt_stddev": 0.0,
        "jitter": 0.0
    },
    "details": [
        {
            "connected_time": "2024-12-01T14:15:11.302364Z",
            "response_time": "2024-12-01T14:15:14.352015Z",
            "rtt": 0,
            "bytes_received": 236,
            "status_msg": "Destination host unreachable",
            "status_code": 3
        },
        {
            "connected_time": "2024-12-01T14:15:14.352273Z",
            "response_time": "2024-12-01T14:15:17.388644Z",
            "rtt": 0,
            "bytes_received": 236,
            "status_msg": "Destination host unreachable",
            "status_code": 3
        }
    ]
}
```

Network Traceroute Test

Trace the path packets take from the source to a specified destination across a computer network, identifying each hop along the way.

- Send packets with incrementally increasing TTL values to trace the network path.
- Identify nodes (routers/switches) based on ICMP "Time Exceeded" messages.
- Determine the path and measure round-trip time to each node.

Requirements

Library	Version
icmplib	3.0.4

Input

```
message TracerouteTestConfig {
  string target_host = 1;
  int32 ttl_max = 2;
  int32 packet_size = 3;
  string timeout = 4;
  int32 repeats = 5;
}
```

Output

The schema of the output is defined as follows:

```
message TracerouteTestResult {
  int32 run_id = 1; // Unique identifier for the test run
  TestStatus status = 2; // Status of the test run
  Summary summary = 3;  // Nested message containing summary information
  repeated Detail details = 4;  // Repeated nested message containing detailed ir
}

message Summary {
  string IP_address = 1;   // Target host IP address
  int32 min_hops = 2;   // Minimum number of hops
  int32 max_hops = 3;  // Maximum number of hops
  double path_stability = 4;  // Path stability, a value between 0 and 1 (0: unst
  double packet_loss = 5;   // Packet loss percentage
}

message Detail {
  int32 run = 1;    // Run number
  repeated Hop hops = 2; // Repeated nested message containing hop information
}

message Hop {
  int32 hop_number = 1;    // Hop number
  string hop_ip = 2;  // IP address of the hop (can be not available)
  double hop_rtt = 3; // Round-trip time to the hop
}
```

How to run simple test

```
sudo pytest monitor_traceroute.py
```

The result will be in the `test` directory in file `output.json` For running the test you need `sudo` permissions just as you would need them for running the script itself.

Examples

INPUT:

```
{
  "target_host": "8.8.8.8",
  "ttl_max" : 50,
  "packet_size": 200,
  "timeout": 3,
  "repeats" : 2
}
```

OUTPUT:

```
{
    "run_id": 1,
    "status": "completed",
    "summary": {
        "IP_address": "8.8.8.8",
        "min_hops": 9,
        "max_hops": 9,
        "path_stability": 1.0,
        "packet_loss": 0.0
    },
    "details": [
        {
            "run": 1,
            "hops": [
                {
                    "hop_number": 1,
                    "hop_ip": "147.229.220.1",
                    "hop_rtt": 4.612
                },
                {
                    "hop_number": 2,
                    "hop_ip": "147.229.254.69",
                    "hop_rtt": 0.462
                },
                {
                    "hop_number": 3,
                    "hop_ip": "147.229.253.233",
                    "hop_rtt": 0.377
                },
                {
                    "hop_number": 4,
                    "hop_ip": "147.229.252.17",
                    "hop_rtt": 1.976
                },
                {
                    "hop_number": 5,
                    "hop_ip": "10.5.0.46",
                    "hop_rtt": 4.756
                },
            {
                "hop_number": 6,
                "hop_ip": "195.113.157.70",
                "hop_rtt": 4.401
            },
            {
                "hop_number": 7,
                "hop_ip": "192.178.99.19",
                "hop_rtt": 4.229
            },
            {
                "hop_number": 8,
                "hop_ip": "216.239.51.183",
                "hop_rtt": 4.158
            },
            {
                "hop_number": 9,
                "hop_ip": "8.8.8.8",
                "hop_rtt": 4.027
            }
        ]
    },
    {
        "run": 2,
        "hops": [
            {
                "hop_number": 1,
                "hop_ip": "147.229.220.1",
                "hop_rtt": 1.634
            },
            {
                "hop_number": 2,
                "hop_ip": "147.229.254.69",
                "hop_rtt": 0.371
            },
            {
                "hop_number": 3,
                "hop_ip": "147.229.253.233",
                "hop_rtt": 0.57
            },
            {
                "hop_number": 4,
                "hop_ip": "147.229.252.17",
                "hop_rtt": 1.584
            },
            {
                "hop_number": 5,
                "hop_ip": "10.5.0.46",
                "hop_rtt": 4.58
            },
            {
                "hop_number": 6,
                "hop_ip": "195.113.157.70",
                "hop_rtt": 4.592
            },
            {
                "hop_number": 7,
                "hop_ip": "192.178.99.19",
                "hop_rtt": 4.19
            },
            {
                "hop_number": 8,
                "hop_ip": "216.239.51.183",
                "hop_rtt": 4.092
            },
            {
                "hop_number": 9,
                "hop_ip": "8.8.8.8",
                "hop_rtt": 4.03
            }
        ]
    }
  ]
}
```

Example (failed test due to unreachable target)

INPUT

```
{
    "target_host": "192.168.1.11",
    "ttl_max": 10,
    "packet_size": 200,
    "timeout": 3,
    "repeats": 1
}
```

OUTPUT

```
{
    "run_id": 1,
    "status": "completed",
    "summary": {
        "IP_address": "192.168.1.11",
        "min_hops": 3,
        "max_hops": 3,
        "path_stability": "Target not reached",
        "packet_loss": 0.0
    },
    "details": [
        {
            "run": 1,
            "hops": [
                {
                    "hop_number": 1,
                    "hop_ip": "147.229.220.1",
                    "hop_rtt": 3.301
                },
                {
                    "hop_number": 2,
                    "hop_ip": "147.229.254.69",
                    "hop_rtt": 0.534
                },
                {
                    "hop_number": 3,
                    "hop_ip": "147.229.253.233",
                    "hop_rtt": 0.707
                }
            ]
        }
    ]
}
```

Network DNS Test

This test is designed to verify the functionality of DNS services by resolving specific DNS names or a list of predefined DNS entries. It performs DNS resolution using standard methods and collects its results. The test aims to provide an overview of the state of the DNS system and detailed results for the individual resolutions, helping in identifying potential issues or verifying the correct operation of DNS services.

Requirements

Library	Version
dnspython	2.5.0

INPUT

```
message DnsResolveTestConfig {
  repeated string query_domains = 1; // List of domains to query
  string query_type = 2; // Type of DNS query (e.g., "A" for address records)
}
```

Parameter	Type	Description
query_domains	List[str]	A list of DNS names to be resolved during the test.
query_type	str	A query type, commonly, "A", "AAAA".
nameservers	List[str]	The list of nameservers to use, if None use the default configuration of the resolver

Output

The output of the test is divided into two main sections: summary information and detailed information.

```
// Message representing the overall test results
message DnsResolveTestResult {
  ID run_id = 1; // The unique identifier of the test run
  TestStatus status = 2; // The overall status of the tests
  Summary summary = 3; // A nested message for the summary of the test results
  repeated Detail details = 4; // An array of detailed results for individual tes
}
```

```
// Summary information of all tests
message Summary {
  int32 total_tests = 1; // The total number of tests conducted
  int32 success_count = 2; // The number of tests that were successful
  int32 failure_count = 3; // The number of tests that failed
  int32 response_time_avg = 4; // The average resolution(response) time across al
  int32 response_time_min = 5; // The minimum resolution(response) time observed
  int32 response_time_max = 6; // The maximum resolution(response) time observed
  string resolution_type = 7; // The type of DNS query performed (e.g., "A", "AA
}
```

```
// Detailed information about each test  if the status is "success"
```

```
message Detail {
  string target_host = 1; // The DNS name queried
  repeated string IP_address = 2; // A list of resolved IP addresses
  int32 expiration_time = 3;  // The number of seconds for which the DNS answer c
  int32 response_time = 4; // The time taken to resolve the DNS name
  string status = 5; // The status of the DNS resolution ("success")
  string status_code = 6; // The response code of the DNS query (e.g., "NOERROR",
  string nameservers_used = 7; // The nameservers used for the resolution
  DS_detail ds = 8; // The delegation signer of the DNS query (only has a value i
  string cname = 9; // The canonical name of the DNS query (only has a value if t
  string ns = 10; // The name server of the DNS query (only has a value if the qu
  SOA_detail soa = 11; // The start of authority of the DNS query (only has a val
  repeated CAA_detail caa = 12; // The certification authority authorization of t
  repeated DNSKEY_detail dnskey = 13; // The DNS key of the DNS query (only has a
  repeated RRSING_detail rrsig = 14; // The resource record signature of the DNS

}

// Detailed information about each test if the status is "failure"
message Detail {
  string target_host = 1; // The DNS name queried
  repeated string IP_address = 2; // A list of resolved IP addresses
  int32 expiration_time = 3;  // The number of seconds for which the DNS answer c
  int32 response_time = 4; // The time taken to resolve the DNS name
  string status = 5; // The status of the DNS resolution ("failed")
  string status_code = 6; // The response code of the DNS query (e.g., "NOERROR",
  string error_message = 7; // The error message describing the failure

}

// Detailed information about the delegation signer (DS) of the DNS query
message DS_detail {
  string key_tag = 1; // The key tag value identifying the DNSKEY record
  string algorithm = 2; // The algorithm used for the key in human-readable form
  string digest_type = 3; // The digest type used by the DS record
  string digest = 4; // The digest value in hexadecimal format
}

// Detailed information about the start of authority (SOA) of the DNS query
message SOA_detail {
  string mname = 1; // The primary name server for the zone
  string rname = 2; // The email address of the responsible person for the zone
  int32 serial = 3; // The serial number of the zone
  int32 refresh = 4; // The time interval before the zone should be refreshed in
  int32 retry = 5; // The time interval before a failed refresh should be retriec
  int32 expire = 6; // The time interval that the zone is valid without refresh i
  int32 minimum = 7; // The minimum TTL value for the zone in seconds
}

// Detailed information about the certification authority authorization (CAA) of
message CAA_detail {
  string flags = 1; // The flags byte of the CAA record as a string
  string tag = 2; // The property tag of the CAA record as a string
  string value = 3; // The property value of the CAA record as a string
}

// Detailed information about the DNS key (DNSKEY) of the DNS query
message DNSKEY_detail {
  string flags = 1; // The flags field of the DNSKEY record as a string
  string protocol = 2; // The protocol field of the DNSKEY record as a string
  string algorithm = 3; // The algorithm used by the DNSKEY record, in human-reac
  string public_key = 4; // The public key associated with the DNSKEY record as a
}

// Detailed information about the resource record signature (RRSIG) of the DNS qu
message RRSIG_detail {
  string type_covered = 1; // The type of resource record covered by the signatu
  string algorithm = 2; // The algorithm used by the RRSIG record, expressed in h
  string labels = 3; // The number of labels in the original RRSIG owner name, pr
  string original_ttl = 4; // The original TTL (Time to Live) value of the covere
  string signature_expiration = 5; // The expiration time of the signature, prese
  string signature_inception = 6; // The inception time of the signature, present
  string key_tag = 7; //  The key tag of the DNSKEY record that generated the sig
  string signer_name = 8; // The domain name of the signer of the RRSIG record, p
  string signature = 9; // The signature value itself, represented in hexadecima
}
```

Summary Information

Field	Type	Description
total_tests	int	The total number of DNS resolutions attempted.
success_count	int	The number of successful DNS resolutions.
failure_count	int	The number of failed DNS resolutions.
response_time_avg	float	The average time taken for DNS resolutions, in milliseconds.
response_time_min	float	The minimum time taken for DNS resolutions, in milliseconds.
response_time_max	float	The maximum time taken for DNS resolutions, in milliseconds.
resolution_type	string	The type of DNS query performed (e.g., "A", "AAAA")

Detailed Information

Each entry in the detailed information section will correspond to an individual DNS resolution attempt and include the following fields:

Field	Type	Description
target_host	str	The DNS name that was resolved.
IP_address	List[str]	The IP addresses returned, if any.
expiration_time	float	The number of seconds for which the DNS answer can be cached and is considered valid
response_time	float	The time taken for the DNS resolution, in milliseconds.

Field	Type	Description
status	str	The status of the DNS resolution. It can be either "success" or "failed".
status_code	str	A DNS response code string. It can be NOERROR, NXDOMAIN, SERVFAIL, ...

For successful resolutions, the following additional fields are included:

Field	Type	Description
ds	DS_detail	The delegation signer of the DNS query.
cname	str	The canonical name of the DNS query.
ns	str	The name server of the DNS query.
soa	SOA_detail	The start of authority of the DNS query.
caa	List[CAA_detail]	The certification authority authorization of the DNS query.
dnskey	List[DNSKEY_detail]	The DNS key of the DNS query.
rrsig	List[RRSIG_detail]	The resource record signature of the DNS query.

For failed resolutions, the following additional field is included:

Field	Type	Description
error_message	str	The error message describing the failure.

Run simple test

```
pytest monitor_dns.py
```

The result will be in the `test` directory in file `output.json`

Examples

INPUT:

```
{
  "target_hosts": ["example.com", "vutbr.cz"],
  "query_type" : "A",
  "nameservers": ["188.116.92.133"]
}
```

OUTPUT:

```
{
    "run_id": 1,
    "status": "completed",
    "summary": {
        "total_tests": 2,
        "successful_tests": 2,
        "failed_tests": 0,
        "response_time_avg": 80.2747,
        "response_time_min": 74.0213,
        "response_time_max": 86.5281,
        "resolution_type": "A"
    },
    "details": [
        {
            "target_host": "example.com.",
            "IP_address": [
                "93.184.215.14"
            ],
            "expiration_time": 2999,
            "response_time": 74.0213,
            "status": "success",
            "status_code": "NOERROR",
            "nameservers_used": [
                "188.116.92.133"
            ],
            "ds": "N/A",
            "cname": "N/A",
            "ns": "N/A",
            "soa": "N/A",
            "caa": "N/A",
            "dnskey": "N/A",
            "rrsig": "N/A"
        },
        {
            "target_host": "vutbr.cz.",
            "IP_address": [
                "147.229.2.90"
            ],
            "expiration_time": 226,
            "response_time": 86.5281,
            "status": "success",
            "status_code": "NOERROR",
            "nameservers_used": [
                "188.116.92.133"
            ],
            "ds": "N/A",
            "cname": "N/A",
            "ns": "N/A",
            "soa": "N/A",
            "caa": "N/A",
            "dnskey": "N/A",
            "rrsig": "N/A"
        }
    ]
}
```

Example (failed test domain does not exist)

INPUT:

```
{
  "query_domains": ["nexistujicidomena.cz"],
  "query_type" : "A",
  "nameservers" : ["188.166.92.133"]
}
```

OUTPUT:

```
{
    "run_id": 1,
    "status": "completed",
    "summary": {
        "total_tests": 1,
        "successful_tests": 0,
        "failed_tests": 1,
        "response_time_avg": 5401.9175,
        "response_time_min": 5401.9175,
        "response_time_max": 5401.9175,
        "resolution_type": "A"
    },
    "details": [
        {
            "target_host": "nexistujicidomena.cz",
            "IP_address": [],
            "expiration_time": "N/A",
            "response_time": 5401.9175,
            "status": "failed",
            "status_code": "TIMEOUT",
            "error_message": "The resolution lifetime expired after 5.402 seconds: Ser
        }
    ]
}
```

NAME

`network.smtp`

VERSION

1.0.0

INFO

SMTP (Simple Mail Transfer Protocol) is a standard communication protocol used for sending and receiving email messages over a network. This test is designed to check if the SMTP service is available on a given `target_host` and attempts to send an email if the service is open.
FYI: when `send_email_flag` is `True`, you should run test by cmd line `pytest -s`.
However, if this test runs across the Internet, it's recommended to set `send_email_flag` to `False` due to security reasons and not letting sensetive data run in the Internet. `starttls()` ensures that email communication between the client and server, but not what is after the server, if there will be redirect of the email.

IMPLEMENTATION NOTES

The script connects to the specified `target_host` through the specified `target_port`. The connection timeout is 5 seconds
If the port number is `25` or `587` it will send firstly operation `starttls()` for security purpose.
Then it sends 3 operations in the following order: `ehlo`, `noop` and `quit`.
`EHLO` -> Extended Hello, it is process of introducing yourself to the server and requesting extended capabilities. You get a list of features and options supported by the server in response.
`NOOP` -> No Operation, it allows to check if the SMTP service is up and if the server is responding without actually sending an email. It's a lightweight way to verify the service's availability.
`QUIT` -> Operation, that ends session between server and client. It ensures that the connection is closed properly.
It also stores the time when the connection was established and, in the case of a NOOP operation, how long it took for the response server to respond.

Used modules

If version is not included - it means it is part of standard Python3 version.

- `smtplib` module (https://docs.python.org/3/library/smtplib.html): Provides SMTP client session object that can be used to send mail to any internet machine with an SMTP or ESMTP listener daemon.
- `email` module (https://docs.python.org/3/library/email.html): is a library for managing email messages.
- `getpass` module (https://docs.python.org/3/library/getpass.html): is a Portable password input. It does not echo the password, when user is prompting.
- `re` module (https://docs.python.org/3/library/re.html): is module that lets using regular expressions.
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to check if a string is a valid hostname (`is_hostname`).

Status code's and it's semantic

I added the most common status code, that you will probably get.
More you can find in - SMTP return codes

- **2yz Positive completion**
    - 211 System status, or system help reply
    - 214 Help message (A response to the HELP command)
    - 220 `domain` Service ready
    - 221 `domain` Service closing transmission channel
    - 221 2.0.0 Goodbye
    - 235 2.7.0 Authentication succeeded

    - 240 QUIT
    - 250 Requested mail action okay, completed
- **5yz Permanent negative completion**
    - 500 Syntax error, command unrecognized
    - 500 5.5.6 Authentication Exchange line is too long
    - 501 Syntax error in parameters or arguments
    - 502 Command not implemented
    - 503 Bad sequence of commands
    - 504 Command parameter is not implemented
    - 504 5.5.4 Unrecognized authentication type
    - 521 Server does not accept mail
    - 523 Encryption Needed
    - 530 5.7.0 Authentication required
    - 534 5.7.9 Authentication mechanism is too weak
    - 535 5.7.8 Authentication credentials invalid
    - 538 5.7.11 Encryption required for requested authentication mechanism
    - 554 5.3.4 Message too big for system
    - 556 Domain does not accept mail

INPUT

Parameter	Type	Description
target_host	string	The hostname or IPv4/6 address
target_port	string	The port number
send_email_flag	string	`True` - send email /// `False` - do not send email

OUTPUT

Field	Type	Description
target_host	string	The hostname or IPv4/6 address
target_port	string	The port number
IP_address	string	Show `target_host` 's IP address
sendmail_op	string	Contains the result of attemting sending email if it set `True`
connected_time	string	Time when the script connected to a specific port
server_msg	string	Time when the script connected to a specific port
ehlo_op	dictionary	Collects the result of sending EHLO operation to server
noop_op	dictionary	Collects the result of sending NOOP operation to server
quit_op	dictionary	Collects the result of sending QUIT operation to server
name_op -> status_code	string	Status code got from server by sending NOOP/QUIT/EHLO operations

Field	Type	Description
name_op -> status_msg	string	Contains messeage result from NOOP/QUIT/EHLO operations
ehlo_op -> SIZE	string	Maximum size of message, that server can handle in MB
response_time	string	Time taken to send and receive the result from the NOOP operation
duration	string	Total time taken to check port health

EXAMPLE

Input Example (JSON format):

```
{
    "target_host": "kazi.fit.vutbr.cz",
    "target_port": "25",
    "send_email_flag": "False"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "kazi.fit.vutbr.cz",
        "target_port": "25",
        "IP_address": "147.229.8.12",
        "connected_time": "2024-11-22T11:58:56.596428Z",
        "server_msg": "250-kazi.fit.vutbr.cz Hello static-84-42-180-82.bb.vodafor
        "ehlo_op": {
            "status_code": "250",
            "status_msg": "['ENHANCEDSTATUSCODES', 'PIPELINING', 'EXPN', 'VERB',
            "SIZE": "32.0 MB"
        },
        "noop_op": {
            "status_code": "250",
            "status_msg": "b'2.0.0 OK'"
        },
        "response_time": "0.019s",
        "quit_op": {
            "status_code": "221",
            "status_msg": "b'2.0.0 kazi.fit.vutbr.cz closing connection'"
        },
        "duration": "0.246s"
    },
    "queue": {
        "target_host": "kazi.fit.vutbr.cz",
        "target_port": "25",
        "IP_address": "147.229.8.12",
        "connected_time": "2024-11-22T11:58:56.596428Z",
        "server_msg": "250-kazi.fit.vutbr.cz Hello static-84-42-180-82.bb.vodafor
        "ehlo_op": {
            "status_code": "250",
            "status_msg": "['ENHANCEDSTATUSCODES', 'PIPELINING', 'EXPN', 'VERB',
            "SIZE": "32.0 MB"
        },
        "noop_op": {
            "status_code": "250",
            "status_msg": "b'2.0.0 OK'"
        },
        "response_time": "0.019s",
        "quit_op": {
            "status_code": "221",
            "status_msg": "b'2.0.0 kazi.fit.vutbr.cz closing connection'"
        },
        "duration": "0.246s"
    }
}
```

EXAMPLE (Failed operation)

Input Example (JSON format):

```
{
    "target_host": "kazi.fit.vutbr.cz123",
    "target_port": "25",
    "send_email_flag": "False"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "kazi.fit.vutbr.cz123",
        "target_port": "25",
        "retcode": "ERROR: on port 25 can't connect.",
        "errcode": "[Errno -2] Name or service not known"
    },
    "queue": {
        "target_host": "kazi.fit.vutbr.cz123",
        "target_port": "25",
        "retcode": "ERROR: on port 25 can't connect.",
        "errcode": "[Errno -2] Name or service not known"
    }
}
```

NAME

`network.imap`

VERSION

1.0.0

INFO

Internet Message Access Protocol, or IMAP, is an Internet standard protocol used by email clients to retrieve email messages from a mail server over a TCP/IP connection.
This test is designed to check the IMAP service on a given server by sending some operations to IMAP server, if the port state is open.

IMPLEMENTATION NOTES

The script connects to the specified `target_host` through the specified `target_port`. The connection timeout is 5 seconds
If the port number is **not** `993`, it will send firstly operation `starttls()` for security purpose.
Then it sends 2 operations in the following order: `noop()` and `capability()`.
`NOOP` -> No Operation, it allows to check if the SMTP service is up and if the server is responding without actually sending an email. It's a lightweight way to verify the service's availability.
`CAPABILITY` -> Operation, that requests for listing the capabilities that the server supports.
It also stores the time when the connection was established and, in the case of a `NOOP` operation, how long it took for the response server to respond.

Used modules

- `imaplib==2.58` module (https://docs.python.org/3/library/imaplib.html): IMAP4 protocol client.
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to check if a string is a valid hostname (`is_hostname`).

INPUT

Parameter	Type	Description
target_host	strings	The hostname or IPv4/6 address
target_port	string	The port number
login_flag	string	If it set "True", then it tries to login to the server
login_username	string	Login credentials for the server
login_server	string	On which server it will login

OUTPUT

Field	Type	Description
IP_address	string	Show `target_host` 's IP address
connected_time	string	Time when the script connected to a specific port
protocol_version	string	Which version of IMAP4 server use.
noop_op	dictionary	Collects the result of sending NOOP operation to server
name_op -> status_code	string	Status code got from server by sending NOOP operation

Field	Type	Description
name_op -> status_msg	string	Contains messeage result from server of NOOP operations
response_time	string	Time taken to send and receive the result from the NOOP operation
server_msg	string	Contains messeage result from server of CAPABILITY operations
duration	string	Total time taken to check port health

EXAMPLE

Input Example (JSON format):

```
{
    "target_host": "kazi.fit.vutbr.cz",
    "target_port": "993",
    "login_flag": "False",
    "login_username": "xassat00@vutbr.cz",
    "login_server": "eva.fit.vutbr.cz"
}
```

Output Example (JSON format):

```
{
    "output": {
        "target_host": "kazi.fit.vutbr.cz",
        "target_port": "993",
        "IP_address": "2001:67c:1220:808::93e5:80c",
        "protocol_version": "IMAP4REV1",
        "connected_time": "2024-11-22T11:27:38.236504Z",
        "noop_op": {
            "status_code": "OK",
            "status_msg": "[b'NOOP completed.']"
        },
        "response_time": "0.019s",
        "server_msg": "('OK', [b'IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
        "duration": "1.471s"
    },
    "queue": {
        "target_host": "kazi.fit.vutbr.cz",
        "target_port": "993",
        "IP_address": "2001:67c:1220:808::93e5:80c",
        "protocol_version": "IMAP4REV1",
        "connected_time": "2024-11-22T11:27:38.236504Z",
        "noop_op": {
            "status_code": "OK",
            "status_msg": "[b'NOOP completed.']"
        },
        "response_time": "0.019s",
        "server_msg": "('OK', [b'IMAP4rev1 SASL-IR LOGIN-REFERRALS ID ENABLE IDLE
        "duration": "1.471s"
    }
}
```

Example (Failed operation)

Input Example (JSON Format)

```
{
    "target_host": "kazi.fit.vutbr.cz123",
    "target_port": "993",
    "login_flag": "False",
    "login_username": "xstudent00@vutbr.cz",
    "login_server": "kazi.fit.vutbr.cz"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "kazi.fit.vutbr.cz123",
        "target_port": "993",
        "retcode": "ERROR: can't connect to target.",
        "errcode": "[Errno -2] Name or service not known"
    },
    "queue": {
        "target_host": "kazi.fit.vutbr.cz123",
        "target_port": "993",
        "retcode": "ERROR: can't connect to target.",
        "errcode": "[Errno -2] Name or service not known"
    }
}
```

NAME

`network.mqtt`

VERSION

1.0.0

INFO

MQTT (Message Queuing Telemetry Transport) - is a lightweight network protocol designed to exchange messages between devices with low bandwidth or high latency. It is ideal for IoT (Internet of Things) and M2M (machine-to-machine) communications. These IoT devices use MQTT for data transmission as it is easy to implement and can efficiently transmit IoT data. MQTT supports the transmission of messages from devices to the cloud and vice versa.

How to test?

Do it using root privileges (su or sudo)

```
1. pip3 install -r requirements.txt
2. pytest test_mqtt.py /// pytest
```

IMPLEMENTATION NOTES

Using `paho-mqtt` library, script creates connection to `target_host` . After that it starts `loop` for **0.3 seconds**, so `client` has time to `subscribe` on some topic and get response from `broker` . After receiving topic `payload` and getting `SUBACK` it ends connection by stopping `loop` and client disconnecting.
For test purpose, I subscribe on some **reserved** topics in `input.json` .

Used modules

- `paho-mqtt==2.1.0` module (https://docs.python.org/3/library/socket.html): A Python implementation of SSHv2..
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to ping servers before testing service.

INPUT

Parameter	Type	Description
target_host	string	The hostname or IP address
topic_names	string_list	The list of topics to obtain information divided by `,`

OUTPUT

Field	Type	Description
IP_address	string	Show `target_host` 's IP address
connected_time	string	Time connected to the service
duration	string	Total time taken to the monitoring
response_time	string	Time taken to send and receive the result
is_connected	bool	Is agent succesfully connected to the service
reason_code_list	string	Result, after sending SUBSCRIBE
$SYS/broker/version	string	Topic name, to subscribe for test purpose
$SYS/broker/uptime	string	Topic name, to subscribe for test purpose

EXAMPLE

Input Example (JSON format):

```
{
    "target_host": "test.mosquitto.org",
    "topic_names": "python/mqtt,$SYS/broker/version,$SYS/broker/uptime"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "test.mosquitto.org",
        "IP_address": "91.121.93.94",
        "is_connected": true,
        "connected_time": "2024-06-29T16:42:43.155399Z",
        "reason_code_list": "[ReasonCode(Suback, 'Granted QoS 0')]",
        "$SYS/broker/version": "b'mosquitto version 2.0.99'",
        "$SYS/broker/uptime": "b'1140410 seconds'"
    },
    "queue": {
        "target_host": "test.mosquitto.org",
        "IP_address": "91.121.93.94",
        "is_connected": true,
        "connected_time": "2024-06-29T16:42:43.155399Z",
        "reason_code_list": "[ReasonCode(Suback, 'Granted QoS 0')]",
        "$SYS/broker/version": "b'mosquitto version 2.0.99'",
        "$SYS/broker/uptime": "b'1140410 seconds'"
    }
}
```

Example (Failed operation)

Input Example (JSON Format)

```
{
    "target_host": "2001:41d0:a:6f1c::12",
    "topic_names": "python/mqtt,$SYS/broker/version,$SYS/broker/uptime"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "status": "error",
        "error": {
            "error_code": "CONFIG FILE ERROR",
            "description": "timed out"
        }
    },
    "queue": {
        "status": "error",
        "error": {
            "error_code": "CONFIG FILE ERROR",
            "description": "timed out"
        }
    }
}
```

NAME

`network.ntp`

VERSION

1.0.0

INFO

`NTP` is designed to synchronize the time throughout an entirenetwork infrastructure, including servers, switches, routers, hostmachines, wireless access points, uninterruptible power supply(UPS), and so on.
This test is designed to check the NTP service on a given server by connecting to NTP server and collecting information.

IMPLEMENTATION NOTES

The script connects to the specified `target_host` . The connection timeout is 5 seconds Script creates client object using `NTPClient()` from `ntplib` and sends request to the NTP server time. All data from the reques are stored in `response` , which is parsed to the `probe` for output.

Used modules

- `ntplib==0.4.0` module (https://pypi.org/project/ntplib/): simple interface to query NTP servers from Python.
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to check if a string is a valid hostname (`is_hostname`).

INPUT

Parameter	Type	Description
target_host	string	The hostname or IP address

OUTPUT

The output is a hostname as key, and dictionary as a value, that contains the results of check. `"someserver": {...}`

Field	Type	Description
target_host	string	The hostname or IP address
IP_address	string	Show `target_host` 's IP address
state	string	Does the server has open/closed/filtered state on a selected port.
connected_time	string	Time when the script connected to a specific port.
delay	string	Round-trip delay to the NTP server.
offset	string	The time difference (in seconds) between the server and the client clocks.
leap_indicator	string	2-bit integer warning of an impending leap second to be inserted or deleted in the last minute of the current month.
stratum	string	The stratum level of the NTP server.
refid	string	32-bit code identifying the particular server or reference clock.
root_delay	string	Total round-trip delay to the reference clock.
root_dispersion	string	Total dispersion to the reference clock.

Field	Type	Description
precision	string	8-bit signed integer representing the precision of the system clock, in `log2` seconds. For instance, a value of `-18` corresponds to a precision of about **one microsecond**.
tx_time	string	Time at the server when the response left for the client.
dest_time	string	Time at the client when the reply arrived from the server.
recv_time	string	Time at the server when the request arrived from the client.

EXAMPLE

Input Example (JSON format):

```
{
    "target_host": "time.apple.com"
}
```

Output Example (JSON format):

```
{
    "output": {
        "target_host": "tik.cesnet.cz",
        "IP_address": "195.113.144.201",
        "connected_time": "2024-11-22T11:58:02.884535Z",
        "delay": "0.021407127380371094",
        "offset": "-10.703453540802002",
        "leap_indicator": "no warning",
        "stratum": "1",
        "refid": "ATOM",
        "root_delay": "0.0",
        "root_dispersion": "0.0009918212890625",
        "precision": "-23",
        "tx_time": "2024-11-22T11:57:52.169491Z",
        "recv_time": "2024-11-22T11:57:52.169408Z",
        "dest_time": "2024-11-22T11:58:02.883648Z"
    },
    "queue": {
        "target_host": "tik.cesnet.cz",
        "IP_address": "195.113.144.201",
        "connected_time": "2024-11-22T11:58:02.884535Z",
        "delay": "0.021407127380371094",
        "offset": "-10.703453540802002",
        "leap_indicator": "no warning",
        "stratum": "1",
        "refid": "ATOM",
        "root_delay": "0.0",
        "root_dispersion": "0.0009918212890625",
        "precision": "-23",
        "tx_time": "2024-11-22T11:57:52.169491Z",
        "recv_time": "2024-11-22T11:57:52.169408Z",
        "dest_time": "2024-11-22T11:58:02.883648Z"
    }
}
```

EXAMPLE (Failed operation)

Input Example (JSON format):

```
{
    "target_host": "time.apple.com123"
}
```

Output Example (JSON format):

```
{
    "output": {
        "target_host": "time.apple.com123",
        "retcode": "ERROR: Can not connect to target the target.",
        "errcode": "[Errno -2] Name or service not known"
    },
    "queue": {
        "target_host": "time.apple.com123",
        "retcode": "ERROR: Can not connect to target the target.",
        "errcode": "[Errno -2] Name or service not known"
    }
}
```

NAME

`network.snmp`

VERSION

1.0.0

INFO

SNMP (Simple Network Management Protocol) – is a lightweight protocol used for managing and monitoring network devices. It is widely utilized for collecting information from devices like routers, switches, servers, and IoT systems. SNMP operates over IP networks, making it ideal for monitoring network performance and detecting issues in real-time. Devices using SNMP can communicate status updates and metrics, such as CPU usage or network traffic, to a central management system. SNMP supports both the querying of data and sending of unsolicited alerts (traps) from devices to management consoles.

IMPLEMENTATION NOTES

This script uses `easysnmp` library to perform SNMP GET test on the target host. It uses input parametrs such as `oids` and `community_string` . It creates prepared session at the beginning, then performs an SNMP GET operation to retrieve a particular piece of information.

This test works with SNMP version 2

Used modules

- `easysnmp==0.2.6` module (https://easysnmp.readthedocs.io/en/latest/#): to work with SNMP service.
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to check if a string is a valid hostname (`is_hostname`).

INPUT

Parameter	Type	Description
target_host	string	The hostname or IP address
oids	string	List of OID for SNMP GET divided by `,`
community_string	string	Community string to get access

OUTPUT

Field	Type	Description
IP_address	string	Show `target_host` 's IP address if it is hostname
oids	dict	Contains `get_value` and `snmp_data_type` .
get_value	string	Contains result of GET operation.
response_time	string	Time taken to send and receive response from the server.
snmp_data_type	string	Contains what data type has the result.
pushed_oid	string	Contains what OID was pushed to the target. Only appears, if there is one OID in `input.json`

EXAMPLE

Input Example (JSON format):

```
{
    "target_host": "localhost",
    "oids": "1.3.6.1.2.1.1.1.0,1.3.6.1.2.1.1.5.0",
    "community_string": "public"
}
```
Output Example (JSON format):
```
{
    "output": {
        "target_host": "isa.fit.vutbr.cz",
        "IP_address": "2001:67c:1220:8b0::93e5:b012",
        "1.3.6.1.2.1.1.5.0": {
            "get_value": "isa.fit.vutbr.cz",
            "snmp_data_type": "OCTETSTR"
        },
        "1.3.6.1.2.1.1.5.1": {
            "get_value": "NOSUCHINSTANCE",
            "snmp_data_type": "NOSUCHINSTANCE"
        },
        "response_time": 0.045
    },
    "queue": {
        "target_host": "isa.fit.vutbr.cz",
        "IP_address": "2001:67c:1220:8b0::93e5:b012",
        "1.3.6.1.2.1.1.5.0": {
            "get_value": "isa.fit.vutbr.cz",
            "snmp_data_type": "OCTETSTR"
        },
        "1.3.6.1.2.1.1.5.1": {
            "get_value": "NOSUCHINSTANCE",
            "snmp_data_type": "NOSUCHINSTANCE"
        },
        "response_time": 0.045
    }
}
```
EXAMPLE (Failed operation)

Input Example (JSON format):
```
{
    "target_host": "localhost",
    "oids": "1.3.6.1.2.1.1.1.0,1.3.6.1.2.1.1.5.0",
    "community_string": "wrong_community_string"
}
```
Output Example (JSON format):
```
{
    "output": {
        "status": "error",
        "error": {
            "error_code": "ERROR: something went wrong during testing",
            "description": "timed out while connecting to remote host"
        }
    },
    "queue": {
        "status": "error",
        "error": {
            "error_code": "ERROR: something went wrong during testing",
            "description": "timed out while connecting to remote host"
            }
        }
    }
```

NAME

`network.ftp`

VERSION

1.0.0

INFO

FTP (File Transfer Protocol) - is an application layer protocol responsible for transferring data between two systems. An FTP connection is created between a client and a server, after which they communicate with each other using the network. To do this, the user can obtain permission by providing credentials to the FTP server or use anonymous FTP.
This script uses an anonymous login method and also tries to connect FTP server over TLS/SSL for security reasons. (FTP server also needs to support TLS/SSL, otherwise it return error code and the connection is shutdown)

IMPLEMENTATION NOTES

Script tries connect to the server in the defined `target_host` . After that, anonymous logging is performed, then the `data connection` is secured using `prot_p()` functions from the same `ftplib` library. Then the script issues `retrlines("LIST")` operation, which is in Linux operation same command `ls -l` . It also calculates time taken to respond from the server. And in the end it sends `quit` operation to ensure correct disconnect from the service.

Used modules

- `ftplib==(no info)` module (https://docs.python.org/3/library/ftplib.html): Provides client side of the FTP protocol. Version
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to check if target host is hostname by function `is_hostname()` .

Status code's and it's semantic

List of all FTP service return codes you can see in RFC959.
This is list of common return codes, that you will see:

- `120` → Service ready in nnn minutes
- `125` → Data connection already open; transfer starting
- `150` → File status okay; about to open data connection
- `230` → User logged in, proceed
- `200` → Command okay
- `226` → Closing data connection / Requested file action successful
- `221` → Service closing control connection
- `332` → Need account for login
- `421` → Service not available, closing control connection
- `425` → Can't open data connection
- `426` → Connection closed; transfer aborted
- `500` → Syntax error or command unrecognized.
- `501` → Syntax error in parameters or arguments
- `502` → Command not implemented
- `530` → Not logged in

INPUT

Parameter	Type	Description
target_host	string	The hostname or IP address

OUTPUT

Field	Type	Description
target_host	string	Target host IPv4/IPv6 address or hostname
connected_time	string	Time when the script connected to a specific port
login_op	string	Anonymously login to the FTP server
welcome_op	string	FTP server **WELCOME** string, that **indicates succesfully connection**
protect_op	string	Turn security on data connection to `private`
retrlines_op	string	Operation to check FTP server functionality
quit_op	string	Disconnect from FTP server
response_time	string	Time taken to send and receive the result from the NOOP operation
duration	string	Total time taken to check port health

In case of error connection. You will see also:

Field	Type	Description
retcode	string	Error messeage code
errmsg	string	Error messeage detail

EXAMPLE

Input Example (JSON format):

```
{
    "target_host": "ftp1.at.proftpd.org"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "ftp.gnu.org",
        "IP_address": "2001:470:142:3::b",
        "connected_time": "2024-11-22T11:26:14.220344Z",
        "welcome_op": "220 GNU FTP server ready.",
        "login_op": "Failed: 530 Please login with USER and PASS.",
        "response_time": "0.118s",
        "quit_op": "221 Goodbye.",
        "duration": "0.536s"
    },
    "queue": {
        "target_host": "ftp.gnu.org",
        "IP_address": "2001:470:142:3::b",
        "connected_time": "2024-11-22T11:26:14.220344Z",
        "welcome_op": "220 GNU FTP server ready.",
        "login_op": "Failed: 530 Please login with USER and PASS.",
        "response_time": "0.118s",
        "quit_op": "221 Goodbye.",
        "duration": "0.536s"
    }
}
```

EXAMPLE (Failed operation)

Input Example (JSON format):

```
{
    "target_host": "192.168.1.11"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "192.168.1.11",
        "retcode": "ERROR:",
        "errcode": "timed out"
    },
    "queue": {
        "target_host": "192.168.1.11",
        "retcode": "ERROR:",
        "errcode": "timed out"
    }
}
```

NAME

`performance.bandwidth`

VERSION

1.0.0

INFO

Network performance is used as an indicator to assess key metrics like throughput, latency, jitter, and packet loss between two endpoints or network agents. Iperf3 is a tool for network performance measurement. It is a cross-platform tool that can produce standardized performance measurements for any network. Iperf3 has client and server functionality, and can create data streams to measure the throughput between the two ends in one or both directions. The data streams can be either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). IPerf3 provides insights into network quality and capacity, making it invaluable for diagnosing network issues, evaluating network upgrades, and optimizing network performance. Test are implimented as a Python script using Iperf3: to measure network performance.

REQUIREMENTS

To install iperf3 on Rocky Linux, run: `yum install iperf3` .

IMPLIMENTATION NOTES

The `peformance.bandwidth.clint.py` script connects to the `peformance.bandwidth.server.py` through a specific `target_host` and `target_port` .
To establish a connection using `iperf3` , the client sends test data to a listening server. Here's a step-by-step process:
`1. Start the Iperf3 Server` - On the server machine run `peformance.bandwidth.server.py` . This script starts the iperf3 server in listening mode on `port 5201` by default (you can specify a different port when needed).
Tests can also be done using public iperf3 serverlists
`2. Start the Iperf3 Client` - On the client machine run `peformance.bandwidth.clint.py` . By default, it connects on port 5201 unless a different port is specified in the Input configuration file.
`3. Connection Established` - The client sends a connection request to the server using TCP or UDP (depending on the test mode). Once connected, the client transmits data to the server, measuring network bandwidth, latency, and other performance metrics.
`4. Default Test Duration` - The test runs for 10 seconds by default, which can be adjusted in the input configuration file.

INPUT

Parameter	Type	Description
`host`	String	Specifies the server IP address or hostname to connect to for the test.
`port`	Integer	Sets the port number on the server to connect to (default is 5201).
`flag`	Flag	Specifies that the test should use UDP instead of TCP.
`duration`	Integer	Defines the duration of the test in seconds (default is 10 seconds).
`reverse`	Flag	Runs the test in reverse mode, measuring the bandwidth from the server to the client.

OUTPUT

Field	Description
`host`	IP address or hostname of the server to which the client is connected.

Field	Description
`port`	Port number used for the connection.
`interval`	Time interval (in seconds) for reporting the metrics, shown for each reported result.
`bytes`	Total number of bytes transferred during the test.
`bitrate`	Calculated throughput in bits per second (bps), indicating the rate of data transfer.
`jitter`	The variation in packet arrival time, measured in milliseconds, relevant for UDP tests.
`loss`	Percentage of packets lost during the test (relevant in UDP mode).
`test_duration`	Duration of the test as specified by the user, or the actual duration if it varied.
`mode`	Indicates whether the test is running in TCP or UDP mode.
`tcp_window_size`	The TCP window size used for the connection, relevant for TCP performance evaluation.
`udp_buffer_size`	The size of the UDP buffer used for sending packets, relevant in UDP mode.

Iperf3 KEY OPTIONS

- **Client Options**
  - -c: - Run in client mode, connecting to the specified server.
  - --bidir: - Perform a bidirectional test (both client and server send data).
  - -R, --reverse: - Run a reverse test (server sends data to client).
  - -p, --port: - Specify the server port (default is 5201).
  - -f, --format : - Format for output (k, m, g, K, M, G for kilo, mega, or giga bits/bytes).
  - -i, --interval : - Report results at regular intervals (default: 1 second).
  - -t, --time : - Test duration in seconds (default: 10 seconds).
  - --bind : - Bind to a specific local IP address.
  - --cport : - Specify the local port for the client.
  - -V, --verbose: - Provide more detailed output.
  - -J: - Output results in JSON format.
  - --logfile : - Write output to a specified log file.
  - -V, --verbose: - Display the iperf3 version and exit.
  - -u, --udp: - Use UDP instead of TCP.
  - --length, -l : - Set length of the buffer to read/write (default: 128 KB for TCP, 8 KB for UDP).
  - --bandwidth [KMG]: - Target bandwidth for UDP tests (default: 1 Mbps).
  - --pacing-timer [KMG]: - Set the interval for UDP packet pacing.
- **Server Options**
  - -s, --server: - Run in server mode.
  - -D, --daemon: - Run the server as a daemon.
  - --logfile : - Write server output to a specified file.
  - --pidfile : - Write the process ID to a specified file.

EXAMPLE

TCP Input Example (JSON Format)

```
{
    "host": "105.235.237.2",
    "port": "5201",
    "protocol": "tcp",
    "duration": "5",
    "reverse": "false",
}
```

TCP Output Example (JSON Format)

```
{
    "start": {
        "connected": [
            {
                "socket": 7,
                "local_host": "192.168.0.132",
                "local_port": 50863,
                "remote_host": "173.214.175.122",
                "remote_port": 5202
            }
        ],
        "version": "iperf 3.17.1",
        "system_info": "Darwin Captain.local 24.0.0 Darwin Kernel Version 24.0.0:
        "timestamp": {
            "time": "Tue, 19 Nov 2024 20:05:32 UTC",
            "timesecs": 1732046732
        },
        "connecting_to": {
            "host": "nyc.speedtest.is.cc",
            "port": 5202
        },
        "cookie": "zj4d34zzn475dapt3q2sxnxzfcbisryw2nrl",
        "tcp_mss_default": 1348,
        "target_bitrate": 0,
        "fq_rate": 0,
        "sock_bufsize": 0,
        "sndbuf_actual": 131072,
        "rcvbuf_actual": 131072,
        "test_start": {
            "protocol": "TCP",
            "num_streams": 1,
            "blksize": 131072,
            "omit": 0,
            "duration": 1,
            "bytes": 0,
            "blocks": 0,
            "reverse": 0,
            "tos": 0,
            "target_bitrate": 0,
            "bidir": 0,
            "fqrate": 0,
            "interval": 1
        }
    },
    "intervals": [
        {
            "streams": [
                {
                    "socket": 7,
                    "start": 0,
                    "end": 1.003706,
                    "seconds": 1.0037059783935547,
                    "bytes": 1179648,
                    "bits_per_second": 9402339.13431934,
                    "omitted": false,
                    "sender": true
                }
            ],
            "sum": {
                "start": 0,
                "end": 1.003706,
                "seconds": 1.0037059783935547,
                "bytes": 1179648,
                "bits_per_second": 9402339.13431934,
                "omitted": false,
                "sender": true
            }
        }
    ],
    "end": {
        "streams": [
            {
                "sender": {
                    "socket": 7,
                    "start": 0,
                    "end": 1.003706,
                    "seconds": 1.003706,
                    "bytes": 1179648,
                    "bits_per_second": 9402338.93191831,
                    "sender": true
                },
                "receiver": {
                    "socket": 7,
                    "start": 0,
                    "end": 1.121017,
                    "seconds": 1.003706,
                    "bytes": 830368,
                    "bits_per_second": 5925819.144580323,
                    "sender": true
                }
            }
        ],
        "sum_sent": {
            "start": 0,
            "end": 1.003706,
            "seconds": 1.003706,
            "bytes": 1179648,
            "bits_per_second": 9402338.93191831,
            "sender": true
        },
        "sum_received": {
            "start": 0,
            "end": 1.121017,
            "seconds": 1.121017,
            "bytes": 830368,
            "bits_per_second": 5925819.144580323,
            "sender": true
        },
        "cpu_utilization_percent": {
            "host_total": 0.825434431194818,
            "host_user": 0.20585323977960565,
            "host_system": 0.6195811914152124,
            "remote_total": 0.0015251656362020422,
            "remote_user": 0,
            "remote_system": 0.0015251656362020422
        },
        "receiver_tcp_congestion": "cubic"
    },
    "host": "nyc.speedtest.is.cc",
    "port": "5202",
    "protocol": "tcp",
    "duration": 1,
    "reverse": false,
    "ttfb": 5.086
}
```

EXAMPLE

UDP Input Example (JSON Format)

```
{
    "host": "105.235.237.2",
    "port": "5201",
    "protocol": "udp",
    "duration": "5",
    "reverse": "false",
}
```

UDP Output Example (JSON Format)

```
{
    "start": {
        "connected": [
            {
                "socket": 7,
                "local_host": "nyc.speedtest.is.cc",
                "local_port": 52755,
                "remote_host": "173.214.175.122",
                "remote_port": 5202
            }
        ],
        "version": "iperf 3.17.1",
        "system_info": "Darwin Captain.local 24.0.0 Darwin Kernel Version 24.0.0:
        "timestamp": {
            "time": "Tue, 19 Nov 2024 20:09:03 UTC",
        "timesecs": 1732046943
    },
    "connecting_to": {
        "host": "nyc.speedtest.is.cc",
        "port": 5202
    },
    "cookie": "u7ih5v4slh7mmrfjemrzkssha32cpiftk3gq",
    "target_bitrate": 1048576,
    "fq_rate": 0,
    "sock_bufsize": 0,
    "sndbuf_actual": 9216,
    "rcvbuf_actual": 786896,
    "test_start": {
        "protocol": "UDP",
        "num_streams": 1,
        "blksize": 1348,
        "omit": 0,
        "duration": 1,
        "bytes": 0,
        "blocks": 0,
        "reverse": 0,
        "tos": 0,
        "target_bitrate": 1048576,
        "bidir": 0,
        "fqrate": 0,
        "interval": 1
    }
},
"intervals": [
    {
        "streams": [
            {
                "socket": 7,
                "start": 0,
                "end": 1.000251,
                "seconds": 1.000251054763794,
                "bytes": 132104,
                "bits_per_second": 1056566.7438856813,
                "packets": 98,
                "omitted": false,
                "sender": true
            }
        ],
        "sum": {
            "start": 0,
            "end": 1.000251,
            "seconds": 1.000251054763794,
            "bytes": 132104,
            "bits_per_second": 1056566.7438856813,
            "packets": 98,
            "omitted": false,
            "sender": true
        }
    }
],
"end": {
    "streams": [
        {
            "udp": {
                "socket": 7,
                "start": 0,
                "end": 1.000251,
                "seconds": 1.000251,
                "bytes": 132104,
                "bits_per_second": 1056566.801732765,
                "jitter_ms": 0.26374305619287425,
                "lost_packets": 0,
                "packets": 98,
                "lost_percent": 0,
                "out_of_order": 0,
                "sender": true
            }
        }
    ],
    "sum": {
        "start": 0,
        "end": 1.117025,
        "seconds": 1.117025,
        "bytes": 132104,
        "bits_per_second": 1056566.801732765,
        "jitter_ms": 0.26374305619287425,
        "lost_packets": 0,
        "packets": 98,
        "lost_percent": 0,
        "sender": true
    },
    "sum_sent": {
        "start": 0,
        "end": 1.000251,
        "seconds": 1.000251,
        "bytes": 132104,
        "bits_per_second": 1056566.801732765,
        "jitter_ms": 0,
        "lost_packets": 0,
        "packets": 98,
        "lost_percent": 0,
        "sender": true
    },
    "sum_received": {
        "start": 0,
        "end": 1.117025,
        "seconds": 1.117025,
        "bytes": 132104,
        "bits_per_second": 946113.1129562902,
        "jitter_ms": 0.26374305619287425,
        "lost_packets": 0,
            "packets": 98,
            "lost_percent": 0,
            "sender": false
        },
        "cpu_utilization_percent": {
            "host_total": 73.90588779348393,
            "host_user": 70.31721742921187,
            "host_system": 3.5884494505138456,
            "remote_total": 0.002036423329442827,
            "remote_user": 0,
            "remote_system": 0.002036910046299768
        }
    },
    "host": "nyc.speedtest.is.cc",
    "port": "5202",
    "protocol": "udp",
    "duration": 1,
    "reverse": false,
    "ttfb": 1.859
}
```

EXAMPLE

Input Example (Failed Operation):

```
{
    "host": "192.168.0.132",
    "port": "5201",
    "protocol": "udp",
    "duration": "5",
    "reverse": "false",
}
```

Output Example (JSON Format):

```
{
    "start": {
        "connected": [],
        "version": "iperf 3.17.1",
        "system_info": "Darwin Captain.local 24.0.0 Darwin Kernel Version 24.0.0: Tue
    },
    "intervals": [],
    "end": {},
    "error": "control socket has closed unexpectedly",
    "host": "192.168.0.132",
    "port": "5201",
    "protocol": "tcp",
    "duration": 1,
    "reverse": false,
    "ttfb": 0.034
}
```

NAME

`webapp.http`

VERSION

1.0.0

INFO

HTTP (Hypertext Transfer Protocol) is an application layer protocol used for transmitting hypermedia documents. This monitor aims to provide a wide range of useful information from a single HTTP(s) endpoint, such as response code, headers and transfer times.

WebSocket support

The monitor will switch to WebSocket mode whenever the input `target` contains a valid WebSocket URI, starting with `ws://` or `wss://`. The monitor will verify whether the specified endpoint provides a WebSocket connection by opening a connection and closing it. Successful monitor run is indicated by `valid_ws` being true in output.

Server-sent Events support

Server-sent events are checked manually by toggling the argument `check_sse`. This will send appropriate headers and checks if the response contains expected headers. Successful monitor run is indicated by `valid_sse` being true in output.
When using the SSE monitor the `body` should be considered as unreliable, as it may or may not have received content from the server.

HTTP/3 support

The libcurl library currently considers the support for HTTP/3 experimental. To utilize this feature, the curl tool has to be built with support for this protocol in mind. While some distributions provide pre-build packages with support already built-in (e.g. Debian), some do not (e.g. RHEL). In case you would like to manually build curl with HTTP/3 support, you can follow the ngtcp2 section in this guide. The process requires to install various 3rd party libraries that provide support for QUIC.
The monitor is able to detect if the current libcurl version is capable of using HTTP/3 and will provide appropriate feedback in case the user tries to use it without support.

Requirements

Library	Usage
pycurl	Compatibility layer for accessing libcurl API used to request HTTP endpoints.
certifi	Used to verify trustworthiness of endpoint certificates.

INPUT

Parameter	Type	Description	Default
target_url	str	URL of HTTP endpoint.	mandato
headers	Dictionary[str,str]	Dictionary of custom HTTP headers to be send with the request. These will overwrite existing headers and not alter unmentioned headers.	{} (no hea

Parameter	Type	Description	Default
http_version	float	Sets a restriction on the version of HTTP that will be used to make the request. If the endpoint doesn't support such version, an error will occur. Possible values are `1.0` , `1.1` , `2.0` , or `0` for any version. The HTTP/3 (option `3.0`) is currently considered too experimental to be included. See HTTP/3 support.	0 (any ve
http_method	str	Sets the HTTP method used in the request. Can be `GET` , `POST` , `PUT` or `OPTIONS` .	GET
http_data	str	Sets the HTTP data being sent with the request. Using this option will automatically set the `Content-Type` header to `application/x-www-form-urlencoded` . When using any other form of data, the header can be changed with the `headers` option.	null (no c
body_flag	boolean	When true includes the whole body response in the `body` output field.	true
follow_redirects	boolean	If true, the monitor will query additional redirect URLs if the response contains them. Each call to separate endpoint due to redirections has its own timeout measurements.	false
ip_version	int	Chooses between IPv4, IPv6 or either for communication with endpoint.	0 (either)
response_values	str	Comma separated list of values in output that will be outputted. Value `all` has a special meaning where it ignores all other list values and outputs everything.	"all" (eve value)
timeout	float	Number of seconds before the connection is dropped in case of no response. This timeout includes all transfer operation (name lookup, TLS, TCP, data transfer, etc.).	60.0
connect_timeout	float	Number of seconds before the connection is dropped in case of no response. This timeout measures the time to establish a connection with the endpoint and is disregarded once the connection is made.	30.0

Parameter	Type	Description	Default
auth_method	str	Selects the desired authentication method. Can be `BASIC` , `DIGEST` or `BEARER` . When using `BASIC` or `DIGEST` , the `username` and `password` arguments must be included. When using `DIGEST` method, the `token` argument must be included.	null (no authentic
username	str	String with username used in authentication. Used with `BASIC` or `DIGEST` authentication method.	null
password	str	String with password used in authentication. Used with `BASIC` or `DIGEST` authentication method.	null
token	str	String with token used in authentication. Used with `BEARER` authentication method.	null
check_sse	boolean	Will check whether the endpoint offers Server-sent Events (SSE). To check for SSE the monitor sends `Accept: text/event-stream` HTTP header and checks for `Content-Type: text/event-stream` in response. This option will set `response_timeout` to 3 seconds so that libcurl has time to get HTTP headers.	false

OUTPUT

Name	Type	Description
target_url	str	Actual URL used in the request.
IP_address	str	Destination IP of endpoint.
port	int	Destination port of endpoint.
status_code	int	HTTP response code.
status_string	str	Short string describing the response code.
http_version	int	HTTP version used in the response.
ttfb	int	Time in seconds from the start until first byte of HTTP data is received.
connection_time	int	Time in seconds of the total transfer duration, including TCP/IP stack communication.
redirect_count	str	Number of redirects followed. Makes sense only with `follow_redirects` enabled.
redirect_url	str	Contains URL of the redirect endpoint, but is empty if the response doesn't contain one.

Name	Type	Description
headers	Dictionary[str,str]	Dictionary of HTTP headers used in the response.
bytes_received	int	Number of bytes downloaded with the response.
download_speed	int	Download speed in bytes per second.
body	str	Decoded body of HTTP response. Included if `body_flag` is True.
ws_key	str	Base64 encoded `Sec-WebSocket-Key` used in WebSocket request. Only returned when using WebSocket.
ws_valid	boolean	True if the header `Sec-WebSocket-Accept` is present and contains valid value. See RFC 6455 Section 1.3 for details. Only returned when using WebSocket.
sse_valid	boolean	True if the argument `check_see` is True and the header `Content-Type` is present and contains `text/event-stream` . Only returned when using Server-sent Events.

EXAMPLES

Generic HTTP

Input

```
{
  "target_url": "fit.vut.cz",
  "follow_redirects": true
}
```

Output

```
{
  "output": {
    "run_id": 1,
    "status": "completed",
    "target_url": "https://www.fit.vut.cz/",
    "IP_address": "147.229.9.65",
    "port": 443,
    "status_code": 200,
    "status_string": "OK",
    "http_version": 2.0,
    "ttfb": 2.234877,
    "connection_time": 2.25671,
    "redirect_count": 2,
    "redirect_url": null,
    "headers": {
      "Date": "Fri, 22 Nov 2024 13:22:26 GMT",
      "Server": "Apache/2.4.62 (FreeBSD) OpenSSL/1.1.1w-freebsd",
      "Location": "https://fit.vut.cz/",
      "Content-Length": "227",
      "Content-Type": "text/html; charset=iso-8859-1",
      "location": "https://www.fit.vut.cz/",
      "content-length": "231",
      "content-type": "text/html; charset=UTF-8",
      "date": "Fri, 22 Nov 2024 13:22:28 GMT",
      "server": "Apache/2.4.62 (FreeBSD) OpenSSL/1.1.1w-freebsd",
      "vary": "Accept-Language,Accept-Encoding",
      "x-ua-compatible": "IE=edge",
      "cache-control": "private, max-age=0, no-cache",
      "pragma": "no-cache",
      "set-cookie": "logoShown=1; path=/; secure; SameSite=lax",
      "strict-transport-security": "max-age=15768000",
      "x-frame-options": "sameorigin",
      "referrer-policy": "same-origin",
      "x-content-type-options": "nosniff"
    },
    "download_size": 87922.0,
    "download_speed": 38960.0,
    "body": "..."
  }
}
```

WebSocket

INPUT

```
{
  "target_url": "wss://demo.piesocket.com/v3/channel_123"
}
```

OUTPUT

```
{
  "output": {
    "run_id": 1,
    "status": "completed",
    "target_url": "https://demo.piesocket.com/v3/channel_123",
    "IP_address": "172.105.59.18",
    "port": 443,
    "status_code": 101,
    "status_string": "Switching Protocols",
    "http_version": 1.1,
    "ttfb": 5.394247,
    "connection_time": 5.394343,
    "redirect_count": 0,
    "redirect_url": null,
    "headers": {
      "Server": "nginx/1.25.2",
      "Date": "Fri, 22 Nov 2024 13:24:37 GMT",
      "Connection": "upgrade",
      "Upgrade": "websocket",
      "Sec-WebSocket-Accept": "2kAMAPKFMYzlHctT5EtiX09B8Mw=",
      "X-Powered-By": "Ratchet/0.4.4"
    },
    "download_size": 32.0,
    "download_speed": 5.0,
    "body": "\u0081\u001a{\"error\":\"Missing apiKey\"}\u0088\u0002\u0003\u00e8",
    "ws_key": "Wmh7aT9KVFlaXmJmUyxFZA==",
    "ws_valid": true
  }
}
```

Server-sent Events

INPUT

```
{
  "target_url": "http://sse.dev/test",
  "check_sse": true
}
```

OUTPUT

```
{
  "output": {
    "run_id": 1,
    "status": "completed",
    "target_url": "http://sse.dev/test",
    "IP_address": "192.248.170.164",
    "port": 80,
    "status_code": 200,
    "status_string": "OK",
    "http_version": 1.1,
    "ttfb": 1.59267,
    "connection_time": 3.00121,
    "redirect_count": 0,
    "redirect_url": null,
    "headers": {
      "Server": "nginx/1.27.2",
      "Date": "Fri, 22 Nov 2024 13:25:09 GMT",
      "Content-Type": "text/event-stream",
      "Transfer-Encoding": "chunked",
      "Connection": "keep-alive",
      "access-control-allow-origin": "*",
      "cache-control": "no-cache"
    },
    "download_size": 83.0,
    "download_speed": 27.0,
    "body": "data: {\"testing\":true,\"sse_dev\":\"is great\",\"msg\":\"It works
    "sse_valid": true
  }
}
```

Invalid request

INPUT

```
{
  "target_url": "localhost"
}
```

OUTPUT

```
{
  "output": {
    "status": "error",
    "error": {
      "error_code": "HTTP_MONITOR_ERROR",
      "description": "Error running HTTP monitor: (7, 'Failed to connect to localhost
    }
  }
}
```

NAME

`webapp.security`

VERSION

1.0.0

INFO

This monitor aims to provide a summarized security monitoring of a given HTTP endpoint. It utilizes that with the use of testssl.sh bash script. The script contains a list of known vulnerabilities, as well as generally known security best practices, which it then uses to discover if the endpoint adheres to them. The result of this monitor is a summarization in a form of grade/score, as well as list of detected findings, which may be used to strengthen any discovered security issues.
The runtime of this monitor is usually around a minute. The time required to monitor every possible vulnerability can be quite lengthy.

License notes

The script `testssl.sh` on which this monitor is built on is licensed under the GPLv2 license. Copy of this license can be found in the LICENSE file. The license allows for copy, distribution and modification of the code. Modifications done to the code have to be released back under GPLv2 license. As the script included is not modified in any way, this shouldn't necessitate public re-release.

Requirements

Library	Usage
openssl >= 1.0	Provides the various cryptographic functionality needed for the monitor. This isn't a Python library but an OS library.

No Python libraries are necessary.

INPUT

Parameter	Type	Description	Default value
target_url	str	URL of HTTP endpoint.	mandatory
connect_timeout	int	Max seconds to wait for TCP socket connection.	None
openssl_timeout	int	Max seconds to wait for openssl connection.	None
script_path	str	Path to the `testssl.sh` script file.	"./script/testssl.sh"

OUTPUT

Name	Type	Description
grade	str	Grade summarizing the quality of the endpoint security. `A` being the best and `F` the worst. A special grading of `T` will be given when the certificate is found to be invalid, and `M` when domain name doesn't match the DNS record.
grade_reasons	list[str]	List of reasons given for the provided grading.
final_score	int	Percentile scoring of the quality, with 100 being the best and 0 the worst.

Name	Type	Description
findings	dict[str,list[Finding]]	Dictionary providing lists of testssl findings, categorized into severity levels of `INFO` , `OK` , `LOW` , `MEDIUM` , `HIGH` and `CRITICAL` . Each finding provides an identifier and explanation of the finding. If the finding contains a known vulnerability, a CVE and CWE identifiers may be included as well.

EXAMPLES

Valid usage

Input

```
{
  "target_url": "https://example.com"
}
```

Output

```
{
  "output": {
    "run_id": 1,
    "status": "completed",
    "grade": "B",
    "grade_reasons": [
      "Grade capped to B. TLS 1.1 offered",
      "Grade capped to B. TLS 1.0 offered",
      "Grade capped to A. HSTS is not offered"
    ],
    "final_score": 91,
    "findings": {
      "INFO": [
        ...
      ],
      "OK": [
        ...
      ],
      "LOW": [
        ...
      ],
      "MEDIUM": [
        {
          "id": "BREACH",
          "severity": "MEDIUM",
          "cve": "CVE-2023-3587",
          "cwe": "CWE-310",
          "finding": "potentially VULNERABLE, gzip deflate HTTP compression detec
        },
        ...
      ]
    }
  }
}
```

Error handling

Input

```
{
  "target_url": "localhost"
}
```

Output

```
{
    "output": {
        "status": "error",
        "error": {
            "error_code": "HTTP_SECURITY_MONITOR_ERROR",
            "description": "Error running HTTP security monitor: testssl.sh returned n
        }
    }
}
```

NAME

`webapp.rest`

VERSION

1.0.0

INFO

REST is an architectural style of stateless web-based application API. It defines a set endpoints which are utilized to access a given resource through HTTP methods (called verbs in REST context).

Matching system

For monitoring of whether the endpoint provides expected data, the monitor implements a system for matching against a provided set or subset of data. For example, the REST endpoint could return following XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <city>San Jose</city>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <state>CA</state>
</root>
```

In such a case the user could provide same or similar set of data into `match_data` input, and the monitor will try to match against it. There's two matching scopes which change the way that the monitor uses the provided data: `full` and `partial` .
With `full` scope, the provided data tries to be matched fully, so that all JSON/XML elements have to be exactly the same. In the previous case, the user would have to provide the whole XML document for the match to pass successfully.
When using `partial` scope, the user needs to provide only a subset of data, which are then matched against the total set of data. For example with the previous case, the user could only provide data in form of `<root><firstName>John</firstName></root>` , and the matching system would pass successfully, because the returned document contains all these elements.

HTTP/3 support

The libcurl library currently considers the support for HTTP/3 experimental. To utilize this feature, the curl tool has to be built with support for this protocol in mind. While some distributions provide pre-build packages with support already built-in (e.g. Debian), some do not (e.g. RHEL). In case you would like to manually build curl with HTTP/3 support, you can follow the ngtcp2 section in this guide. The process requires to install various 3rd party libraries that provide support for QUIC.
The monitor is able to detect if the current libcurl version is capable of using HTTP/3 and will provide appropriate feedback in case the user tries to use it without support.

Requirements

Library	Usage
pycurl	Compatibility layer for accessing libcurl API used to request HTTP endpoints.
certifi	Used to verify trustworthiness of endpoint certificates.

INPUT

This monitor utilized the implementation of webapp.http for making HTTP calls, the inputs are therefore fully compatible. The following tables showcase both types of available parameters in this monitor:

REST parameters

Parameter	Type	Description	Default value
match_type	str	Selects what kind of serialization method the endpoint responds with. Can be one of: `json` , `xml` .	`json`
match_scope	str	Selects how to match the provided matching data in `match_data` . Can be one of: `full` , `partial` . For more info on these, see Matching system	`full`
match_data	str	Provides the data to match against. This argument is mandatory if `match_type` or `match_scope` are provided.	null

HTTP parameters

Parameter	Type	Description	Default
target_url	str	URL of HTTP endpoint.	mandato
headers	Dictionary[str,str]	Dictionary of custom HTTP headers to be send with the request. These will overwrite existing headers and not alter unmentioned headers.	{} (no hea
http_version	float	Sets a restriction on the version of HTTP that will be used to make the request. If the endpoint doesn't support such version, an error will occur. Possible values are `1.0` , `1.1` , `2.0` , or `0` for any version. The HTTP/3 (option `3.0`) is currently considered too experimental to be included. See HTTP/3 support.	0 (any ve
http_method	str	Sets the HTTP method used in the request. Can be `GET` , `POST` , `PUT` or `OPTIONS` .	GET
http_data	str	Sets the HTTP data being sent with the request. Using this option will automatically set the `Content-Type` header to `application/x-www-form-urlencoded` . When using any other form of data, the header can be changed with the `headers` option.	null (no c
follow_redirects	boolean	If true, the monitor will query additional redirect URLs if the response contains them. Each call to separate endpoint due to redirections has its own timeout measurements.	false
ip_version	int	Chooses between IPv4, IPv6 or either for communication with endpoint.	0 (either)
response_values	str	Comma separated list of values in output that will be outputted. Value `all` has a special meaning where it ignores all other list values and outputs everything.	"all" (eve value)
timeout	float	Number of seconds before the connection is dropped in case of no	60.0

Parameter	Type	Description	Default
		response. This timeout includes all transfer operation (name lookup, TLS, TCP, data transfer, etc.).	
connect_timeout	float	Number of seconds before the connection is dropped in case of no response. This timeout measures the time to establish a connection with the endpoint and is disregarded once the connection is made.	30.0
auth_method	str	Selects the desired authentication method. Can be `BASIC` , `DIGEST` or `BEARER` . When using `BASIC` or `DIGEST` , the `username` and `password` arguments must be included. When using `DIGEST` method, the `token` argument must be included.	null (no authentic
username	str	String with username used in authentication. Used with `BASIC` or `DIGEST` authentication method.	null
password	str	String with password used in authentication. Used with `BASIC` or `DIGEST` authentication method.	null
token	str	String with token used in authentication. Used with `BEARER` authentication method.	null
check_sse	boolean	Will check whether the endpoint offers Server-sent Events (SSE). To check for SSE the monitor sends `Accept: text/event-stream` HTTP header and checks for `Content-Type: text/event-stream` in response. This option will set `response_timeout` to 3 seconds so that libcurl has time to get HTTP headers.	false

OUTPUT

Name	Type	Description
match	bool	Set to true if the matching arguments are used and the provided data is matched successfully.
http_outputs	dict	Outputs from webapp.http.

HTTP OUTPUT

Name	Type	Description
target_url	str	Actual URL used in the request.
IP_address	str	Destination IP of endpoint.

Name	Type	Description
port	int	Destination port of endpoint.
status_code	int	HTTP response code.
status_string	str	Short string describing the response code.
http_version	int	HTTP version used in the response.
ttfb	int	Time in seconds from the start until first byte of HTTP data is received.
connection_time	int	Time in seconds of the total transfer duration, including TCP/IP stack communication.
redirect_count	str	Number of redirects followed. Makes sense only with `follow_redirects` enabled.
redirect_url	str	Contains URL of the redirect endpoint, but is empty if the response doesn't contain one.
headers	Dictionary[str,str]	Dictionary of HTTP headers used in the response.
bytes_received	int	Number of bytes downloaded with the response.
download_speed	int	Download speed in bytes per second.
body	str	Decoded body of HTTP response. Included if `body_flag` is True.
ws_key	str	Base64 encoded `Sec-WebSocket-Key` used in WebSocket request. Only returned when using WebSocket.
ws_valid	boolean	True if the header `Sec-WebSocket-Accept` is present and contains valid value. See RFC 6455 Section 1.3 for details. Only returned when using WebSocket.
sse_valid	boolean	True if the argument `check_see` is True and the header `Content-Type` is present and contains `text/event-stream` . Only returned when using Server-sent Events.

EXAMPLES

JSON with POST

Input

```
{
  "target_url": "https://httpbin.org/post",
  "http_method": "POST",
  "follow_redirects": true,
  "match_data": "{\"headers\": {\"Accept\": \"*/*\", \"Host\": \"https://httpbin.
  "match_scope": "partial"
}
```

Output

```
{
  "output": {
    "run_id": 1,
    "status": "completed",
    "match": true,
    "http_outputs": {
      "run_id": 1,
      "status": "completed",
      "target_url": "https://httpbin.org/post",
      "IP_address": "52.20.148.183",
      "port": 443,
      "status_code": 200,
      "status_string": "OK",
      "http_version": 2.0,
      "ttfb": 1.9285,
      "connection_time": 1.928623,
      "redirect_count": 0,
      "redirect_url": null,
      "headers": {
        "date": "Fri, 22 Nov 2024 13:17:34 GMT",
        "content-type": "application/json",
        "content-length": "455",
        "server": "gunicorn/19.9.0",
        "access-control-allow-origin": "*",
        "access-control-allow-credentials": "true"
      },
      "download_size": 455.0,
      "download_speed": 235.0,
      "body": "{\n  \"args\": {}, \n  \"data\": \"\", \n  \"files\": {}, \n  \"fo
    }
  }
}
```

XML with GET

INPUT

```
{
  "target_url": "https://mocktarget.apigee.net/xml",
  "http_method": "GET",
  "follow_redirects": true,
  "match_type": "XML",
  "match_data": "<root><city>San Jose</city></root>",
  "match_scope": "partial"
}
```

OUTPUT

```
{
  "output": {
    "run_id": 1,
    "status": "completed",
    "match": true,
    "http_outputs": {
      "run_id": 1,
      "status": "completed",
      "target_url": "https://mocktarget.apigee.net/xml",
      "IP_address": "35.227.194.212",
      "port": 443,
      "status_code": 200,
      "status_string": "OK",
      "http_version": 2.0,
      "ttfb": 0.175937,
      "connection_time": 0.177186,
      "redirect_count": 0,
      "redirect_url": null,
      "headers": {
        "x-powered-by": "Apigee",
        "access-control-allow-origin": "*",
        "x-frame-options": "ALLOW-FROM RESOURCE-URL",
        "x-xss-protection": "1",
        "x-content-type-options": "nosniff",
        "content-type": "application/xml; charset=utf-8",
        "content-length": "141",
        "etag": "W/\"8d-oqSmr/xiG8D5GJ3RBUhqY00xvcA\"",
        "date": "Fri, 22 Nov 2024 13:20:26 GMT",
        "via": "1.1 google",
        "alt-svc": "h3=\":443\"; ma=2592000,h3-29=\":443\"; ma=2592000"
      },
      "download_size": 141.0,
      "download_speed": 795.0,
      "body": "<?xml version=\"1.0\" encoding=\"UTF-8\"?> <root><city>San Jose</city><
    }
  }
}
```

NAME

`security.ssh`

VERSION

1.0.0

INFO

SSH (Secure SHell) - is an application layer network protocol designed for secure remote access to `UNIX` systems. This protocol is effective in that it encrypts all information transmitted over the network. By default, `port 22` is used. It is mainly used for remote management of user data on the server, running service commands, working in console mode with databases.

IMPLEMENTATION NOTES

The script connects to the specified `target_host` through the specified `target_port`. The connection timeout is 5 seconds
Using `socket` library, script creates connection to `target_host` and receives data from the connected socket. Here is usefull data is `SSH Version/Banner` which represents, that service is working and users can connect to it. After that, script closes connection by `socket.close()`.

Used modules

- `socket` module (https://docs.python.org/3/library/socket.html): A Python implementation of SSHv2.
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to ping servers before testing service.

INPUT

Parameter	Type	Description
target_host	string	The hostname or IPv4/6 address

OUTPUT

Field	Type	Description
IP_address	string	Show `target_host`'s IP address
connected_time	string	Time connected to the service
response_time	string	Time taken to get response from server
ssh_banner	string	SSH Version, which indicates postive connection
duration	string	Total time taken for procedure

EXAMPLE

Input Example (JSON format):

```
{
    "target_host": "eva.fit.vutbr.cz"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "eva.fit.vutbr.cz",
        "IP_address": "147.229.176.14",
        "connected_time": "2024-06-29T16:45:19.157495Z",
        "ssh_banner": "SSH-2.0-OpenSSH_9.7\r\n",
        "response_duration": "0.064s",
        "connection_duration": "0.111s"
    },
    "queue": {
        "target_host": "eva.fit.vutbr.cz",
        "IP_address": "147.229.176.14",
        "connected_time": "2024-06-29T16:45:19.157495Z",
        "ssh_banner": "SSH-2.0-OpenSSH_9.7\r\n",
        "response_duration": "0.064s",
        "connection_duration": "0.111s"
    }
}
```

EXAMPLE (Unsuccessful operation)

Input Example (JSON format):

```
{
    "target_host": "192.168.1.11"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "status": "error",
        "error": {
            "error_code": "CONFIG FILE ERROR",
            "description": "local variable 'sock' referenced before assignment"
        }
    },
    "queue": {
        "status": "error",
        "error": {
            "error_code": "CONFIG FILE ERROR",
            "description": "local variable 'sock' referenced before assignment"
        }
    }
}
```

Security TLS/SSL Test

This test is designed to test the initial handshake of TLS (Transport Layer Security) communication. It attempts to create a secured connection to the target host using configurable parameters (TLS version, cipher suites, etc.), collects server parameters, and measures performance metrics such as handshake duration and success rate. The purpose is to ensure that the target host supports desired TLS configurations and performs adequately under various conditions.

Requirements

Library	Version
pyOpenSS	24.1.0
scapy	2.5.0
cryptography	41.0.5

The test needs to be run with elevated privileges to capture packets and perform the handshake.

Input

For TLS version 1.3, the specified cipher suites are ignored, because the library does not support the configuration of cipher suites for TLS 1.3. The library uses the default cipher suites for TLS 1.3.

```
message TLSHandshakeTestConfig {
    string target_host = 1; // The target host to connect to
    int32 target_port = 2; // The target port to connect to
    string tls_version = 3; // The TLS version to use for the connection
    repeated string cipher_suites = 4; // The list of cipher suites to use for th
    repeated string elliptic_curves = 5; // The list of elliptic curves to use fo
    repeated Extension extensions = 6; // The list of extensions to use for the c
    int32 timeout = 7; // The timeout for the connection in seconds
}
```

Parameter	Type	Description
target_host	str	The target host to connect to.
target_port	int	The target port to connect to.
tls_version	str	The TLS version to use for the connection.
cipher_suites	List[str]	The list of cipher suites to use for the connection.
elliptic_curves	List[str]	The list of elliptic curves to use for the connection.
extensions	List[Extension]	The list of extensions to use for the connection.
timeout	int	The timeout for the connection in seconds.

Output

The output of the test is divided into two main sections: summary information and detailed information.

```
// Message representing the overall test results
message TLSHandshakeTestResult {
    ID run_id = 1; // The unique identifier of the test run
    TestStatus status = 2; // The overall status of the tests
    int32 handshake_time = 3; // The time taken to complete the handshake in mill
    string IP_address = 4; // The target IP that was connected to
    int32 target_port = 5; // The target port that was connected to
    string tls_version = 6; // The TLS version used for the connection
    string cipher_suite = 7; // The cipher suite used for the connection
    string elliptic_curve = 8; // The elliptic curve used for the connection
    string SNIs = 9; // The server name indication used for the connection
    string alpn = 10; // The application layer protocol negotiation used for the
    int32 client_extension_count = 11; // The number of client extensions used fo
    int32 server_extension_count = 12; // The number of server extensions used fo
    repeated string client_extension_names = 13; // The names of the client exter
    repeated string server_extension_names = 14; // The names of the server exter
    Extensions extensions = 15; // The extensions used for the connection
    repeated ServerCertificate server_cert_chain = 16; // The server certificate
    ServerCertificate server_cert = 17; // The server certificate information
}

// Message representing the server certificate information
ServerCertificate {
    string subject_cn = 1; // The common name of the subject
    string subject_on = 2; // The organization name of the subject
    string subject_country = 3; // The country of the subject
    string issuer_cn = 4; // The common name of the issuer
    string issuer_on = 5; // The organization name of the issuer
    string issuer_country = 6; // The country of the issuer
    string not_before = 7; // The not before date of the certificate
    string not_after = 8; // The not after date of the certificate
    int64 serial_number = 9; // The serial number of the certificate
    int32 version = 10; // The version of the certificate
    string signature_algorithm = 11; // The signature algorithm of the certificat
    int32 public_key_length = 12; // The public key size of the certificate
    string fingerprint = 13; // The fingerprint of the certificate
}

// Message representing the extensions used for the connection
Extensions {
    string key_usage = 1; // The key usage extension
    string extended_key_usage = 2; // The extended key usage extension
    repeated string authority_info_access = 3; // The authority information acces
}
```

In case of successful handshake, the following fields are included:

Field	Type	Description
run_id	int	The unique identifier of the test run.
status	TestStatus	The overall status of the tests.
handshake_time	int	The time taken to complete the handshake in milliseconds.
IP_address	str	The target IP that was connected to
target_port	int	The target port to connect to.
tls_version	str	The TLS version used for the connection.

Field	Type	Description
cipher_suite	str	The cipher suite used for the connection.
elliptic_curve	str	The elliptic curve used for the connection.
SNIs	str	The server name indication used for the connection.
alpn	str	The application layer protocol negotiation used for the connection.
client_extension_count	int	The number of client extensions used for the connection.
server_extension_count	int	The number of server extensions used for the connection.
client_extension_names	List[str]	The names of the client extensions used for the connection.
server_extension_names	List[str]	The names of the server extensions used for the connection.
extensions	Extensions	The extensions used for the connection.
server_cert_chain	List[ServerCertificate]	The server certificate chain information.
server_cert	ServerCertificate	The server certificate information.

For failed handshake, the following fields are included:

Field	Type	Description
run_id	int	The unique identifier of the test run.
status	TestStatus	The overall status of the tests.
error_message	str	The error message describing the failure.
error_description	str	The description of the error.

How to run simple test

```
sudo pytest monitor_tls.py
```

The result will be in the `test` directory in file `output.json` For running the test you need `sudo` permissions just as you would need them for running the script itself.

Examples

INPUT:

```
{
    "target_host": "www.example.com",
    "target_port": 443,
    "tls_version": "TLSv1.2",
    "cipher_suites": [
        "ECDHE-ECDSA-AES256-GCM-SHA384",
        "ECDHE-RSA-AES256-GCM-SHA384"
    ],
    "elliptic_curves": [
        "prime256v1"
    ],
    "extensions": [
        {
            "type": "sni",
            "data": "example.com"
        },
        {
            "type": "alpn",
            "data": ["spdy/2", "http/1.1"]
        }
    ],
    "timeout": 10
}
```

OUTPUT:

```
{
    "run_id": 1,
    "status": "completed",
    "handshake_time": 459,
    "IP_address": "93.184.215.14",
    "target_port": 443,
    "tls_version": "TLSv1.2",
    "cipher_suite": "ECDHE-RSA-AES256-GCM-SHA384",
    "elliptic_curve": "prime256v1",
    "SNIs": "example.com",
    "alpn": "http/1.1",
    "client_extension_count": 8,
    "server_extension_count": 5,
    "client_extension_names": [
        "TLS Extension - Server Name",
        "TLS Extension - Supported Point Format",
        "TLS Extension - Supported Groups",
        "TLS Extension - Session Ticket",
        "TLS Extension - Application Layer Protocol Negotiation",
        "TLS Extension - Encrypt-then-MAC",
        "TLS Extension - Extended Master Secret",
        "TLS Extension - Signature Algorithms"
    ],
    "server_extension_names": [
        "TLS Extension - Renegotiation Indication",
        "TLS Extension - Supported Point Format",
        "TLS Extension - Session Ticket",
        "TLS Extension - Application Layer Protocol Negotiation",
        "TLS Extension - Extended Master Secret"
    ],
    "addition_server_cert_info": {
        "key_usage": "Digital Signature, Key Encipherment",
        "extended_key_usage": "TLS Web Server Authentication, TLS Web Client Aut
        "authority_info_access": [
            "OCSP - URI:http://ocsp.digicert.com",
            "CA Issuers - URI:http://cacerts.digicert.com/DigiCertGlobalG2TLSRSA
        ]
    },
    "server_cert_chain": [
        {
            "subject_cn": "www.example.org",
            "subject_on": "Internet Corporation for Assigned Names and Numbers",
            "subject_country": "US",
            "issuer_cn": "DigiCert Global G2 TLS RSA SHA256 2020 CA1",
            "issuer_on": "DigiCert Inc",
            "issuer_country": "US",
            "not_before": "2024-01-30 00:00:00",
            "not_after": "2025-03-01 23:59:59",
            "serial_number": 9781292415466404211737309641897402759,
            "version": 3,
            "signature_algorithm": "sha256WithRSAEncryption",
            "public_key_length": 2048,
            "fingerprint": "EF:BA:26:D8:C1:CE:37:79:AC:77:63:0A:90:F8:21:63:A3:D6
        },
        {
            "subject_cn": "DigiCert Global G2 TLS RSA SHA256 2020 CA1",
            "subject_on": "DigiCert Inc",
            "subject_country": "US",
            "issuer_cn": "DigiCert Global Root G2",
            "issuer_on": "DigiCert Inc",
            "issuer_country": "US",
            "not_before": "2021-03-30 00:00:00",
            "not_after": "2031-03-29 23:59:59",
            "serial_number": 17226682543955925492517929723242541158,
            "version": 3,
            "signature_algorithm": "sha256WithRSAEncryption",
            "public_key_length": 2048,
            "fingerprint": "C8:02:5F:9F:C6:5F:DF:C9:5B:3C:A8:CC:78:67:B9:A5:87:B5
        }
    ],
    "server_cert": {
        "subject_cn": "www.example.org",
        "subject_on": "Internet Corporation for Assigned Names and Numbers",
        "subject_country": "US",
        "issuer_cn": "DigiCert Global G2 TLS RSA SHA256 2020 CA1",
        "issuer_on": "DigiCert Inc",
        "issuer_country": "US",
        "not_before": "2024-01-30 00:00:00",
        "not_after": "2025-03-01 23:59:59",
        "serial_number": 9781292415466404211737309641897402759,
        "version": 3,
        "signature_algorithm": "sha256WithRSAEncryption",
        "public_key_length": 2048,
        "fingerprint": "EF:BA:26:D8:C1:CE:37:79:AC:77:63:0A:90:F8:21:63:A3:D6:89:
    }
}
```

Example (failed test because of not existing domain)

INPUT

```
{
    "target_host": "neexistujicidomena.cz",
    "target_port": 443,
    "tls_version": "TLSv1.2",
    "cipher_suites": [
        "ECDHE-ECDSA-AES256-GCM-SHA384",
        "ECDHE-RSA-AES256-GCM-SHA384"
    ],
    "elliptic_curves": [
        "prime256v1"
    ],
    "extensions": [
        {
            "type": "sni",
            "data": "example.com"
        },
        {
            "type": "alpn",
            "data": ["spdy/2", "http/1.1"]
        }
    ],
    "timeout": 6
}
```

OUTPUT

```
{
    "status": "error",
    "error": {
        "error_code": "TLS_TEST_ERROR",
        "description": "Error running TLS test: 'target_host'"
    }
}
```

Example (failed test because of unsupported TLS version)

INPUT

```
{
    "target_host": "www.example.com",
    "target_port": 443,
    "tls_version": "TLSv1.0",
    "cipher_suites": [
        "ECDHE-ECDSA-AES256-GCM-SHA384",
        "ECDHE-RSA-AES256-GCM-SHA384"
    ],
    "elliptic_curves": [
        "prime256v1"
    ],
    "extensions": [
        {
            "type": "sni",
            "data": "example.com"
        },
        {
            "type": "alpn",
            "data": ["spdy/2", "http/1.1"]
        }
    ],
    "timeout": 6
}
```

OUTPUT

```
{
    "run_id": 1,
    "status": "error",
    "error": {
        "error_msg": "SSL Error",
        "description": "[('SSL routines', '', 'no protocols available')]"
    }
}
```

NAME

`security.ldap`

VERSION

1.0.0

INFO

LDAP (Lightweight Directory Access Protocol) - is a lightweight protocol for accessing data organised in a hierarchical structure (directory). It allows you to manage and search information about users, devices, and other resources on your network. It is most often used for authentication and account management in corporate networks.

IMPLEMENTATION NOTES

This script uses `ldap3` library to establish connection to the target host. As an input parametr used only `target_host` .
It creates server object for LDAP connection, then performs `bind` command to the `target_host` . If connection is established successfuly - it sends simple `search` query to check if LDAP service working properly. The result of `search` query can be empty list `[]` or contain some entities. The empty list `[]` does not mean, that something is wrong. The case is to check LDAP service and get response, which `[]` is.

Used modules

- `socket` module (https://docs.python.org/3/library/socket.html): provides access to the BSD socket interface.
- `ldap3==2.9.1` module (https://ldap3.readthedocs.io/en/latest/): used to check ldap service.
- `icmplib==3.0.4` module (https://pypi.org/project/icmplib/): used to check hostname address.

INPUT

Parameter	Type	Description
target_host	string	The hostname or IPv4/6 address

OUTPUT

Field	Type	Description
target_host	string	Target hostname or IPv4/6 address
IP_address	string	IPv4/6 address if it is hostname
connected_time	string	Established connection time
duration	string	Total time taken to operation.
response_time	string	Server response time in milliseconds
search_query	string	Simple query to check LDAP service. It can be empty list or contain some entries.

EXAMPLE

Successful operation [without entries]
Input Example (JSON format):

```
{
    "target_host": "db.debian.org"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "db.debian.org",
        "IP_address": "2001:41b8:202:deb:1a1a:0:52c3:4b6a",
        "connected_time": "2024-10-24T12:22:46.826487Z",
        "resposne_time_ms": 0.19,
        "search_query": []
    },
    "queue": {
        "target_host": "db.debian.org",
        "IP_address": "2001:41b8:202:deb:1a1a:0:52c3:4b6a",
        "connected_time": "2024-10-24T12:22:46.826487Z",
        "resposne_time_ms": 0.19,
        "search_query": []
    }
}
```

Successful operation [with entries]

Input Example (JSON format):

```
{
    "target_host": "ldap.fit.vutbr.cz"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "target_host": "ldap.fit.vutbr.cz",
        "IP_address": "147.229.9.22",
        "connected_time": "2024-10-24T12:21:43.066462Z",
        "resposne_time_ms": 0.01,
        "search_query": [
            "DN: dc=vutbr,dc=cz - STATUS: Read - READ TIME: 2024-10-24T14:21:43.
            <omitted>
        ]
    },
    "queue": {
        "target_host": "ldap.fit.vutbr.cz",
        "IP_address": "147.229.9.22",
        "connected_time": "2024-10-24T12:21:43.066462Z",
        "resposne_time_ms": 0.01,
        "search_query": [
            "DN: dc=vutbr,dc=cz - STATUS: Read - READ TIME: 2024-10-24T14:21:43.
            <omitted>
        ]
    }
}
```

Unsuccessful operation

Input Example (JSON format):

```
{
    "target_host": "kazi.fit.vutbr.cz"
}
```

Output Example (JSON Format)

```
{
    "output": {
        "status": "error",
        "error": {
            "error_code": "ERROR: something went wrong during testing",
            "description": "('unable to open socket', [(LDAPSocketOpenError('socket co
        }
    },
    "queue": {
        "status": "error",
        "error": {
            "error_code": "ERROR: something went wrong during testing",
            "description": "('unable to open socket', [(LDAPSocketOpenError('socket co
        }
    }
}
```

Other SQL DB Test

This test is designed to test the connection to various SQL databases such as PostgreSQL, MySQL, Oracle and Microsoft SQL Server. It measures the time taken to establish a connection, upload and download data, and execute a complex query. You can also specify a query to execute on the database and get the result of the query with the time taken to execute it. The purpose is to ensure that the database is accessible and performs adequately under various conditions.
Part of this test are also scripts that can be used to create the necessary tables and data in the databases for testing purposes and to run the corresponding databases in docker containers for testing purposes. The scripts are located in the `scripts` directory. You need to have Docker installed on your machine to run the databases in docker containers. For each database type is one specific script. Please run the scripts from the root directory of the test. Like this:

```
./scripts/postgresql.sh
```

Requirements

Library	Version
oracledb	2.2.1
pyodbc	5.0.1
mysql-connector-python	8.4.0
psycopg2-binary	2.9.9
pymssql	2.3.0
docker	7.1.0

If the library `psycopg2-binary` does not work, you can try to install `psycopg2==2.9.9` instead.

How to install docker

On `Rocky Linux` just run these commands (for all these steps be the root user):
Optional step:

```
su root
```

1.

```
sudo dnf config-manager --add-repo https://download.docker.com/linux/rhel/docker-
```

2.

```
sudo dnf -y install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

3.

```
sudo systemctl start docker
sudo systemctl enable docker
```

Now you can run the example scripts, that are provided in the folder `scritps/` .
Each dokcer container needs this amount of space:

Database	Size	Version
PostgreSQL	419MB	13
MySQL	594MB	8.4

Database	Size	Version
Oracle	2.91GB	21
MSSQL	1.58GB	2022-latest

Input

```
message SQLDBTestInput {
    string db_type = 1; // The type of the database to connect to
    string target_host = 2; // The host of the database
    int32 target_port = 3; // The port of the database
    string dbname = 4; // The name of the database
    string user = 5; // The username to connect to the database
    string password = 6; // The password to connect to the database
    string query = 7; // The query to execute on the database
}
```

Parameter	Type	Description
db_type	string	The type of the database to connect to (e.g. postgresql, mssql, mysql, oracle).
target_host	string	The host of the database.
target_port	int	The port of the database.
dbname	string	The name of the database.
user	string	The username to connect to the database.
password	string	The password to connect to the database.
query	string	The query to execute on the database (Can be empty, so no query will be executed).

Output

```
// Message representing the overall test results
message SQLDBTestOutput {
    ID run_id = 1; // The unique identifier of the test run
    TestStatus status = 2; // The overall status of the tests
    double connection_time = 3; // The time taken to establish the connection in
    int32 upload_time = 4; // The time taken to upload data in milliseconds
    int32 download_time = 5; // The time taken to download data in milliseconds
    double upload_size = 6; // The size of the uploaded data in MB
    double download_size = 7; // The size of the downloaded data in MB
    double complex_query = 8; // The time taken to execute a complex query in sec
    string query_result = 9; // The result of the query specified in the input (
    double query_time = 10; // The time taken to execute the query in millisecon
}
```

In case of successful connection, the following fields are included:

Field	Type	Description
run_id	int	The unique identifier of the test run.
status	TestStatus	The overall status of the tests.
connection_time	int	The time taken to establish the connection in milliseconds.
upload_time	int	The time taken to upload data in milliseconds.

Field	Type	Description
download_time	int	The time taken to download data in milliseconds.
upload_size	double	The size of the uploaded data in MB.
download_size	double	The size of the downloaded data in MB.
complex_query	double	The time taken to execute a complex query in seconds.
query_result	str	The result of the query specified in the input.
query_time	double	The time taken to execute the query in milliseconds.

For failed handshake, the following fields are included:

Field	Type	Description
run_id	int	The unique identifier of the test run.
status	TestStatus	The overall status of the tests.
error_message	str	The error message describing the failure.
error_description	str	The description of the error.

Automatic tests

You can run the automatic tests for this test by running the following command in the root directory of the test:

```
pytest monitor_nosql.py
```

This will create output files in the `test` directory with the results of the tests and named `<db_name>_output.json` .
Or if you want to you can run the the databases separately in docker containers. You can do this by running the following scripts in the `scripts` directory:

```
./scripts/mongo.sh
```

But the output will be only printed to the console.

Examples

INPUT:

```
{
    "db_type" : "postgresql",
    "target_host" : "0.0.0.0",
    "target_port" : 5432,
    "dbname" : "testdb",
    "user" : "postgres",
    "password" : "mysecretpassword",
    "query" : ""
}
```

OUTPUT:

```
{
    "run_id": 1,
    "status": "success",
    "data": {
        "connection_time": 5.5094,
        "upload_time": 148.3335,
        "download_time": 2.2187,
        "upload_size": 0.9536,
        "download_size": 0.9536,
        "complex_query": 70.5195,
        "query_result": null,
        "query_time": null
    }
}
```

Example (failed test due to the database not being available)

INPUT:

```
{
    "db_type" : "postgresql",
    "target_host" : "0.0.0.0",
    "target_port" : 5432,
    "dbname" : "testdb",
    "user" : "postgres",
    "password" : "mysecretpassword",
    "query" : ""
}
```

OUTPUT:

```
{
    "status": "error",
    "data": {
        "error_code": "SQL_TEST_ERROR",
        "description": "Error during running test: (20009, b'DB-Lib error message 2000
    }
}
```

Other NOSQL DB Test

This test is designed to test the connection to various NOSQL databases such as MongoDB, Redis, Cassandra and Dynamo. It measures the time taken to establish a connection, upload and download of random data. You can also specify a query to execute on the database and get the result of the query with the time taken to execute it. The purpose is to ensure that the database is accessible and performs adequately under various conditions.
Part of this test are also scripts that can be used to create the necessary tables and data in the databases for testing purposes and to run the corresponding databases in docker containers for testing purposes. The scripts are located in the `scripts` directory. You need to have Docker installed on your machine to run the databases in docker containers. For each database type is one specific script. Please run the scripts from the root directory of the test. Like this:

```
./scripts/mongo.sh
```

Each dokcer container needs this amount of space:

Database	Size	Version
MongoDB	782MB	7.0.15
Redis	117MB	7.2.6
Cassandra	358MB	4.1.0
amazon/dynamodb-local	494MB	2.5.2

Requirements

Library	Version
boto3	1.34.143
cassandra-driver	3.29.1
pymongo	4.8.0
redis	5.0.7
docker	7.1.0

Input

```
message NOSQLDBTestInput {
    string db_type = 1; // The type of the database to connect to
    string target_host = 2; // The host of the database
    int32 target_port = 3; // The port of the database
    string dbname = 4; // The name of the database
    string username = 5; // The username to connect to the database
    string password = 6; // The password to connect to the database
    string query = 7; // The query to execute on the database
    string region = 8; // The region of the database (e.g. us-east-1. Only for dy
}
```

Parameter	Type	Description
db_type	string	The type of the database to connect to (e.g. mongo, cassandra, redis, dynamo).
target_host	string	The host of the database.
target_port	int	The port of the database.

Parameter	Type	Description
dbname	string	The name of the database.
username	string	The username to connect to the database.
password	string	The password to connect to the database.
query	string	The query to execute on the database (Can be empty, so no query will be executed).

Output

```
// Message representing the overall test results
message SQLDBTestOutput {
    ID run_id = 1; // The unique identifier of the test run
    TestStatus status = 2; // The overall status of the tests
    double connection_time = 3; // The time taken to establish the connection in
    int32 upload_time = 4; // The time taken to upload data in milliseconds
    int32 download_time = 5; // The time taken to download data in milliseconds
    double upload_size = 6; // The size of the uploaded data in MB
    double download_size = 7; // The size of the downloaded data in MB
    double query_time = 8; // The time taken to execute the query in milliseconds
    string query_result = 9; // The result of the query specified in the input ((
}
```

In case of successful connection, the following fields are included:

Field	Type	Description
run_id	int	The unique identifier of the test run.
status	TestStatus	The overall status of the tests.
connection_time	int	The time taken to establish the connection in milliseconds.
upload_time	int	The time taken to upload data in milliseconds.
download_time	int	The time taken to download data in milliseconds.
upload_size	double	The size of the uploaded data in MB.
download_size	double	The size of the downloaded data in MB.
query_time	double	The time taken to execute the query in milliseconds.
query_result	str	The result of the query specified in the input.

For failed handshake, the following fields are included:

Field	Type	Description
run_id	int	The unique identifier of the test run.
status	TestStatus	The overall status of the tests.
error_code	str	The error message describing the failure.
description	str	The description of the error.

Automatic tests

You can run the automatic tests for this test by running the following command in the root directory of the test:

```
pytest monitor_nosql.py
```

This will create output files in the `test` directory with the results of the tests and named `<db_name>_output.json` .
Or if you want to you can run the the databases separately in docker containers. You can do this by running the following scripts in the `scripts` directory:

```
./scripts/mongo.sh
```

But the output will be only printed to the console.

Examples

INPUT:

```
{
    "db_type" : "mongodb",
    "host" : "0.0.0.0",
    "port" : 27017,
    "dbname" : "testdb",
    "user" : "mongouser",
    "password" : "mysecretpassword",
    "query" : ""
}
```

OUTPUT:

```
{
    "run_id": 1,
    "status": "success",
    "data": {
        "connection_time": 3.937244415283203,
        "upload_time": 276.87788009643555,
        "download_time": 2.707242965698242,
        "upload_size": 0.95367431640625,
        "download_size": 0.95367431640625
    }
}
```

Other input examples

For **mongodb** the query needs to be in the following format:

```
{
    "table_name": "name_of_the_collection",
    "filter": "..."
}
```

For **dynamodb** the query needs to be in the following format:

~~~
{
    "table_name": "name_of_the_table",
    "key_condition_expression": "...",
    "expression_attribute_values": "..."
}

```json
{
    "db_type" : "mongodb",
    "host" : "0.0.0.0",
    "port" : 27017,
    "dbname" : "testdb",
    "user" : "mongouser",
    "password" : "mysecretpassword",
~~~

```
    "query" : {
        "table_name": "myCollection",
        "filter": "{\"name\" : \"Alice\"}"
        }
}

{
    "db_type" : "redis",
    "host" : "0.0.0.0",
    "port" : 6379,
    "dbname" : "testdb",
    "user" : "default",
    "password" : "mysecretpassword",
    "query" : "SET name 'Alice'"
}

{
    "db_type" : "cassandra",
    "host" : "0.0.0.0",
    "port" : 9042,
    "dbname" : "testdb",
    "user" : "cassandrauser",
    "password" : "mysecretpassword",
    "query" : "SELECT * FROM users WHERE user_id = '12345'"
}

{
    "db_type": "dynamodb",
    "target_host": "0.0.0.0",
    "target_port": 8000,
    "dbname": "testdb",
    "username": "fakeAccessKeyId",
    "password": "fakeSecretAccessKey",
    "region": "us-west-2",
    "query": {
        "table_name": "users",
        "key_condition_expression": "#id = :id",
        "expression_attribute_names": {
            "#id": "id"
        },
        "expression_attribute_values": {
            ":id": {"S": "12345"}
        }
    }
}
```

Example (failed test due to wrong connection parameters)

INPUT:

```
{
    "db_type" : "cassandra",
    "target_host" : "0.0.0.0",
    "target_port" : 9043,
    "dbname" : "testdb",
    "username" : "cassandrauser",
    "password" : "mysecretpassword",
    "query" : ""
}
```

OUTPUT:

```
{
    "run_id": 1,
    "status": "error",
    "data": {
        "error_code": "NOSQL_TEST_ERROR",
        "description": "Error 111 connecting to 0.0.0.0:6379. Connection refused."
    }
}
```