# SNMP MIB Parser.

(11 July 2025)

# Overview

This all started with an accompanying [blog](https://github.com/georgelza/DataPipeline-SNMP_Flink_Fluss), where we poll SNMP agent/s using either a GET or a WALK method using a [Apache Flink SNMP Source Connector](https://github.com/georgelza/SNMP-Flink-Source-connector.git) that we created.

The SNMP Agent polled will respond with either a numeric [OID](https://en.wikipedia.org/wiki/Object_identifier) and data or a text representation and data.

For the numeric value we want to enrich the returned data with the contents found in a MIB file to give it more meaning to the user.

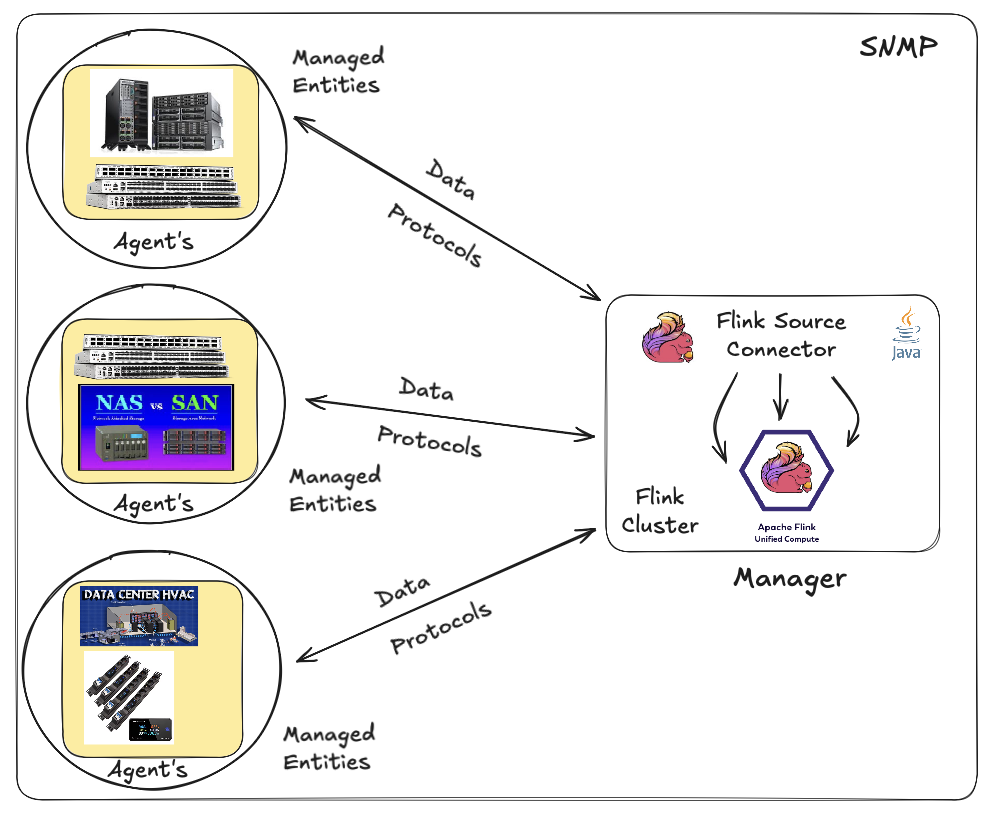
As our main project inserts the returned data into a table, we want to also insert the [SNMP OID](https://en.wikipedia.org/wiki/Object_identifier) (Object Identifier) detail into a table to allow the user to join the 2 sets of records…

Firstly, just a quick explanation, what is SNMP (Simple Network Management Protocol), just in case you don’t know.

As per Wikipedia

[**Simple Network Management Protocol**](https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol)(**SNMP**) is an [Internet Standard](https://en.wikipedia.org/wiki/Internet_Standard)protocol for collecting and organizing information about managed devices on [IP](https://en.wikipedia.org/wiki/Internet_Protocol" \o "Internet Protocol)networks and for modifying that information to change device behavior. Devices that typically support SNMP include [cable modems](https://en.wikipedia.org/wiki/Cable_modem), [routers](https://en.wikipedia.org/wiki/Router_(computing)), [network switches](https://en.wikipedia.org/wiki/Network_switch), servers, workstations, printers, and more

At the simplest level it’s a protocol allowing Datacenter devices, called Agents to send key/value sets of data to what’s called a Network Management Service, aka our manager.



Now what is considered an agent… well, it’s really an open-ended question… It’s probably one of the most underappreciated standards available but underutilized in data centers still today. One I think needs more attention especially based on the rate that data centers are growing, at an amazing rate.

Back to agents, pretty much everything electronic today (bar mobile phones) is enabled as a snmp agent, even your home printer.

Below are 2 examples executed in my home network.

*snmpget -v1 -c abfr24 172.16.10.24 .1.3.6.1.2.1.1.5.0*

*SNMPv2-MIB::sysName.0 = STRING: vaultx*

*snmpwalk -v1 -c abfr24 172.16.10.3 1.3.6.1.2.1.1.2*

*SNMPv2-MIB::sysDescr.0 = STRING: Linux USWAggregation 3.18.24 #0 Thu Aug 30 12:10:54 2018 mips*

*SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10*

*DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1980275900) 229 days, 4:45:59.00*

*SNMPv2-MIB::sysContact.0 = STRING: Default*

*SNMPv2-MIB::sysName.0 = STRING: USWAggregation*

*SNMPv2-MIB::sysLocation.0 = STRING: Stud*

Ok, basic overview done.

The above information returned from the agent can be decoded into a data packet as per below.

oid: 1.3.6.1.2.1.1.2

object\_name: sysObjectID.0

info: sysUpTimeInstance

type: Timeticks

oid\_type: INTEGER32

module: UBNT-MIB

Our [Python](https://www.python.org/) based utility will extract the above data components (there are normally many records/entries) from a MIB file and then allow the user to insert the data into either a [PostgreSQL](https://www.postgresql.org/), [MySQL](https://www.mysql.com/) or [REDIS](https://redis.io/) data store.

You can start by navigating to the root folder of the project and look at the README.md, it will explain how to configure the [Python](https://www.python.org/) Virtual environment, install the required libraries as listed in the requirements file and then execute the utility.

Below is examples of the various database options/tables and how to execute the utility using the supplied shell script.

The following is the expected table structure for [PostgreSQL](https://www.postgresql.org/):

CREATE TABLE IF NOT EXISTS public.snmp\_oid\_metadata (

oid\_string VARCHAR(255) PRIMARY KEY -- The numerical OID (e.g., ".1.3.6.1.2.1.1.1.0")

,object\_name VARCHAR(255) -- The human-readable name (e.g., "sysDescr")

,data\_type VARCHAR(50) -- The data type (e.g., "DisplayString", "Integer32")

,info VARCHAR(2000) -- The textual description from the MIB

,oid\_type VARCHAR(255) -- "scalar", "table", "notification", etc.

,mib\_module VARCHAR(50) -- Source file / Module Name

);

The following is the expected table structure for [MySQL](https://www.mysql.com/):

CREATE TABLE IF NOT EXISTS snmp.snmp\_oid\_metadata (

oid\_string VARCHAR(255) PRIMARY KEY -- The numerical OID (e.g., ".1.3.6.1.2.1.1.1.0")

,object\_name VARCHAR(255) -- The human-readable name (e.g., "sysDescr")

,data\_type VARCHAR(50) -- The data type (e.g., "DisplayString", "Integer32")

,info VARCHAR(2000) -- The textual description from the MIB

,oid\_type VARCHAR(255) -- "scalar", "table", "notification", etc.

,mib\_module VARCHAR(50) -- Source file / Module Name

);

For [REDIS](https://redis.io/) the data is inserted into the Datastore as a key:value pair. The Key is a concatenation of the string: ‘oid:’ and the oid value. The value packaged as JSON payload.

See below example:

Redis Key: oid:.1.3.6.1.2.1.1.1.0

Redis Value:

{

"object\_name": "sysDescr",

"data\_type": "DisplayString",

"info": "A textual description of the entity.",

"oid\_type": "Scalar"

"mib\_module": "TRUENAS-MIB"

}

To execute the Utility see the following script examples:

[MySQL](https://www.mysql.com/):

python3 mib\_parser.py \

--mib-file mibslocal/TRUENAS-MIB.mib \

--mib-dirs mibstd/ \

--db-type mysql \

--db-host localhost \

--db-port 3306 \

--db-user dbadmin \

--db-password dbpassword \

--db-name snmp \

--db-schema snmp \

--tbl-name snmp\_oid\_metadata

[PostgreSQL](https://www.postgresql.org/):

python3 mib\_parser.py \

--mib-file mibslocal/TRUENAS-MIB.mib \

--mib-dirs mibstd/ \

--db-type postgresql \

--db-host localhost \

--db-port 5433 \

--db-user dbadmin \

--db-password dbpassword \

--db-name snmp \

--db-schema public \

--tbl-name snmp\_oid\_metadata

[REDIS](https://redis.io/):

python3 mib\_parser.py \

--mib-file mibslocal/TRUENAS-MIB.mib \

--mib-dirs mibstd/ \

--db-type redis \

--db-host localhost \

--db-port 6379 \

--db-name 0 \

--redis-key-prefix oid



# And In Summary.

Not much to say… this is a larger component of our SNMP Data pipeline. Just one of our building blocks.

Hope it’s of benefit to someone or interesting.

The next idea is brewing already ;) As always.

Good luck, this is all fraught with rabbit holes, as always, so many and you can disappear so easily… but then that’s ½ the fun.



*Note: to execute this blog start with README.md located in the root folder and work from there, it will tell you exactly what to execute in which order to download all the dependencies and build everything. If you have any problems, welcome to reach out to me via one of the below profiles.*

**About Me**

I’m a techie, a technologist, always curious, love data, have for as long as I can remember always worked with data in one form or the other, Database admin, Database product lead, data platforms architect, infrastructure architect hosting databases, backing it up, optimizing performance, accessing it. Data data data… it makes the world go round.

In recent years, pivoted into a more generic Technology Architect role, capable of full stack architecture.

[George Leonard](https://www.linkedin.com/in/george-leonard-945b502/)

[georgelza@gmail.com](mailto:georgelza@gmail.com)

<https://www.linkedin.com/in/george-leonard-945b502/>

<https://medium.com/@georgelza>