**Changes in sonic telemetry to support Multi-Asic**

This document currently lists down design/code changes needed to support telemetry Dial-In mode (gNMI Server) for Multi-ASIC platforms. It covers both DB based and Non-DB based data sets.

1. Need to update go module db\_config.go. This file reads database\_config.json and set the properties

Of Database (eg: retrieve the Redis Unix Socket path and Redis TCP { IPAddress, Port} Bind Port from the file as needed for setting up Unix/TCP connection to Redis Server

* + For Multi-Asic platforms we have database\_global\_config.json that points to database\_config.json of each asic/namespace DB. We need to update to read from each of this files and update utility API to provide DB information for each asic/namespace. Eg:

GetDbSock(namespace)

* + This change will be similar to what we have done in swsscommon (C++ Library) and sonic-py-swsssdk (Python Library)
  + We will need to add utility API to this module to get list of namespace GetDbNamespaceList() and some more as done in C++/Python library
  + We will need to update init() of db\_client.go to connect to each ASIC Redis Server

<https://github.com/Azure/sonic-telemetry/blob/201911/sonic_db_config/db_config.go#L117>

**2 Handling of DB-based Data Sets**

* Current we have two type of DB-based Data Sets:
  1. DB Set that Virtual to Real Path Mapping (v2R). As of now below are the one supported.

"COUNTERS\_DB"/"COUNTERS"/"Ethernet\*"

COUNTERS\_DB"/"COUNTERS"/"Ethernet\*"/"\*"

"COUNTERS\_DB"/"COUNTERS"/"Ethernet\*"/"Queues"

"COUNTERS\_DB"/"COUNTERS"/"Ethernet\*"/"Pfcwd"

* 1. All the other DB based data set. Example:

“STATE\_DB”/PROCESS\_STATS”

* When the GNMI Client send GET/Set/Subscribe request to the server it consists of two parts

(prefix, path).

Prefix 🡺 DB Name to access (DB name is the 1st element in path slice)

Path 🡺 Consists of {TblIdx (2nd element), KeyIdx (3rd element), FieldIdx (4th element) }

Eg: python /gnxi/gnmi\_cli\_py/py\_gnmicli.py -g -t 10.3.147.142 -p 50051 -m get -x COUNTERS/Ethernet0/SAI\_PORT\_STAT\_IF\_IN\_OCTETS -xt COUNTERS\_DB -o "ndastreamingservertest"

**2.1 Current Design of Data Set having V2R mapping and proposed changes.**

* + Real path is define as *type tablePath struct* { dbName, tableName, tableKey } where
    - Db\_name 🡺 prefix (1st element)
    - tableName 🡺 TblIdx (2nd element)
    - tableKey 🡺 This is derived from KeyIdx (3rd element)

<https://github.com/Azure/sonic-telemetry/blob/201911/sonic_data_client/db_client.go#L67>

The 3rd element in request is always <Sonic Interface Name eg: Ethernet0> for all the current Data Set. This is converted into tableKey based as below.

|  |  |  |
| --- | --- | --- |
| Virtual Field | Mapping used for V2R | Example |
| Ethernet\* | COUNTERS\_PORT\_NAME\_MAP | Ethernet100: oid:0x100000000001b |
| Ethernet\*/Queue | COUNTERS\_QUEUE\_NAME\_MAP | Ethernet88:16: oid:0x1500000000041b |

The oid return above is used as tableKey and tuple of (db\_name, tableName, tableKey/oid) is used as final key to access Redis DB

<https://github.com/Azure/sonic-telemetry/blob/201911/sonic_data_client/virtual_db.go#L266>

* + For Multi-asic platforms set of Ethertnet\* port belong to different asic/redis db. So to create V2R mapping we will loop on each asic db and fetch the COUNTERS\_PORT\_NAME\_MAP and COUNTERS\_QUEUE\_NAME\_MAP data

<https://github.com/Azure/sonic-telemetry/blob/201911/sonic_data_client/virtual_db.go#L87>

* + Also, on multi-asic platforms the oid return is not unique across DB’s i.e two different Ethernet port on different ASIC will have same OID.

To create the unique oid we will add asic namespace also as part of table path.

*type tablePath struct* { dbName, tableName, asicNamespace, tableKey }

For single asic platfrom it will be empty (‘’) string and for multi-asic platforms it will point to namespace eg: asic0, asic1,..asicn

* + ‘asicNamespace’ will be used to access correct Redis Server to fetch the data.

Currently Resdis server instance is stored in 1-D map {dbanme:redisInstance} and this will be changed 2-D map {asicNamespace:{dbname: redisInstance}}

var Target2RedisDb make(map[string]map[string]\*redis.Client)

<https://github.com/Azure/sonic-telemetry/blob/201911/sonic_data_client/db_client.go#L334>

* All the API’s in db\_client.go that interface with Redis server will need just single line update to get correct redis server

redisDb:=Target2RedisDb[tblPath.asicNamespace][tblPath.dbName]

<https://github.com/Azure/sonic-telemetry/blob/201911/sonic_data_client/db_client.go#L537>

**2.2 Current Design of Data Set do not have V2R mapping and proposed changes.**

* + For all the dataset that do not have V2R mapping the (prefix,path) is directly used as key
  + To support multi asic for this case proposing to pass asic name space also in target prefix. Namespace will suffix the DbName so to maintain backward compatibility.

Eg: python /gnxi/gnmi\_cli\_py/py\_gnmicli.py -g -t 10.3.147.142 -p 50051 -m get -x PR BGP\_INFO -xt STATE\_DB/ASIC0 -o "ndastreamingservertest"

* + We will retrieve the asic namespace from gnmiPath Request and will be used to retrieve the corresponding Redis Client.
  + If the “/ASICX” is not present, then request will map to default namespace (existing behavior will be achieved
  + API isTargetDb() need to be update to understand new suffix
  + Passing Asic Namespace as Suffix for Data Set that have V2R mapping will be thrown error. This can be enhanced if we need to validate V2R mapping against asic namespace but that will result in lot of cases to handle.

**3. Non-DB based Data-Set**

This are defined when prefix target is “OTHERS’ in client request.

Eg: cpu Info. No change is required for Multi-asic