Open for All.

OCP GLOBAL SUMMIT

Developer`s Overview of Sonic Part 2

To Become a Sonic Developer

1.) https://azure.github.io/SONiC/

2.) User Guide or https://github.com/Azure/SONiC/wiki/Architecture

3.) Developer`s overview Session(s).
Part 1: https://github.com/Azure/SONiC/blob/master/doc/ocp/201903-SONIC/workshop/Developer's%20Overview%20of%20SONiC%20-%20LNKD.pdf
Part 2:

Praveen Chaudhary, SW Engg.

pchaudhary@linkedin.com

1.) Redis-server, different DBs, config with multi Redis.

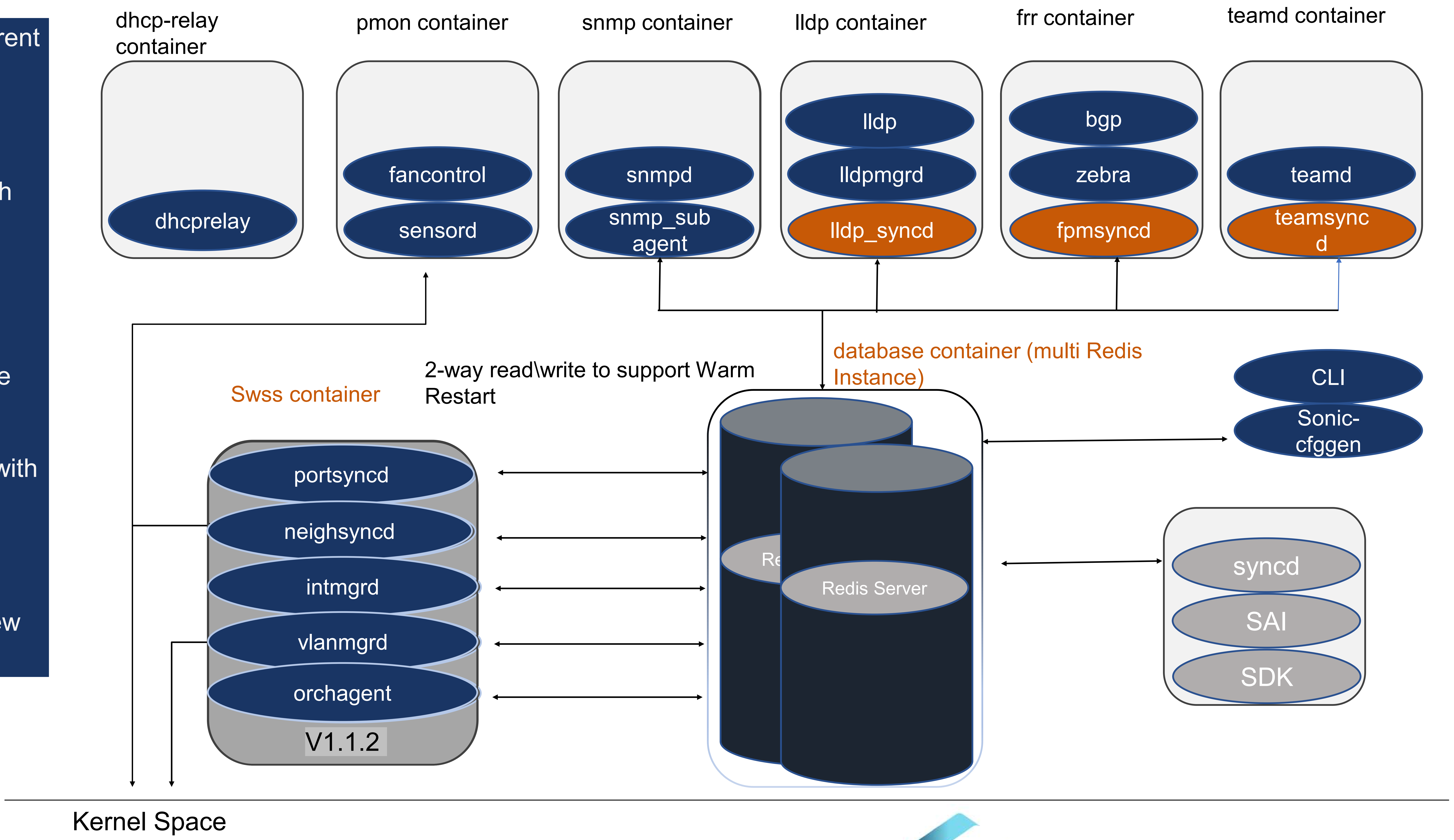
2.) Basic Libraries to interact with Redis with multi Redis.

3.) Component Interaction.

4.) Warm Restart Base Classes.

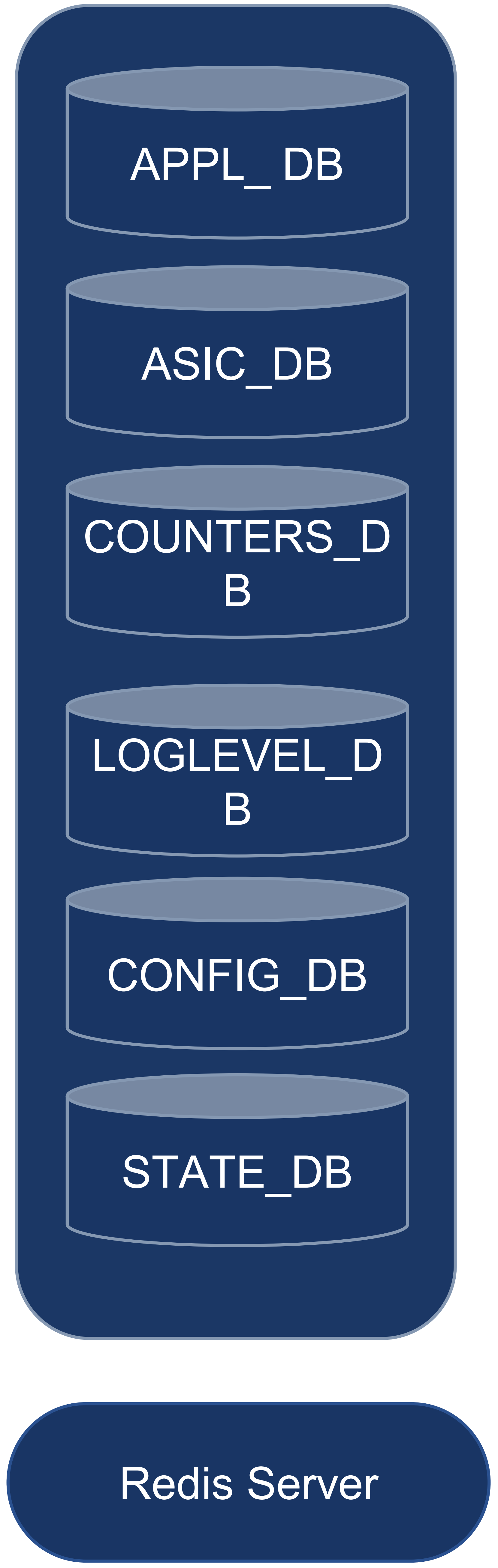
5.) Syncd Processes with warm restart. FRR as special case.

6.) Orchagent Overview with warm restart.



https://github.com/Azure/sonic-swss-common/blob/master/common/schema.h
https://github.com/Azure/sonic-swss/blob/master/doc/swss-schema.md

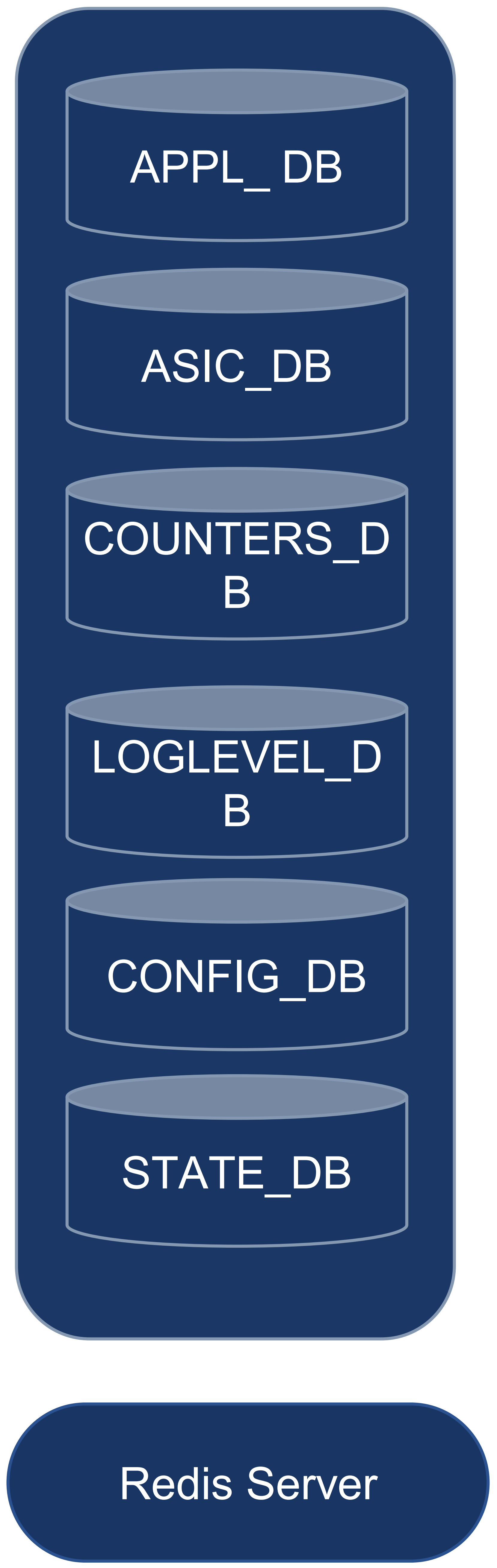
DB name	DB No.	Description	Additional Information
APPL_DB (Application DB)	0	ARP/NDP Entries, BGP Routes, LLDP entries, Next-Hop etc.	"ROUTE_TABLE:10.xxx.79.xxx/26" "INTF_TABLE:Ethernet19:fe80::xxxx:xxxx:feba:xxxx/64" "NEIGH_TABLE:Vlanxxx:10.xxx.10.xxx" "VLAN_MEMBER_TABLE:Vlanxxx:Ethernet29" "PORT_TABLE:Ethernet74" "COPP_TABLE:trap.group.XXX.xxx,lacp" "LLDP_ENTRY_TABLE:Ethernet2"
ASIC_DB	1	Running ASIC Configuration and ASIC State Data.	"ASIC_STATE:SAI_OBJECT_TYPE_ROUTE_ENTRY:{\"dest\":\"xxxx:f3g5:60:4::12b/128\",\"switch_id\":\"oid:0x27770000000000\",\"vr\":\"oid:0x3000000000042\"}" "ASIC_STATE:SAI_OBJECT_TYPE_NEXT_HOP_GROUP_MEMBER:oid:0x2d000000003363" "ASIC_STATE:SAI_OBJECT_TYPE_NEIGHBOR_ENTRY:{\"ip\":\"xxxx:f349:40:a794::2\",\"rif\":\"oid:0x6000000000add\",\"switch_id\":\"oid:0x21000000000000\"}"
COUNTERS_ DB	2	Counter data for port, lag, queue, ACLs.	COUNTERS: CRM:ACL_STATS:INGRESS:LAG:
LOGLEVEL_ DB	3	Log level control for Sonic subsystems	swssloglevel –p buffermgrd NOTICE fpmsyncd NOTICE intfmgrd NOTICE intfsyncd NOTICE neighsyncd NOTICE orchagent NOTICE portsyncd NOTICE syncd NOTICE teamsyncd NOTICE vlanmgrd NOTICE
CONFIG_DB	4	DB for Sonic Configuration	/etc/sonic/config_db.json $config reload\load [sonic-cfggen] **WARM_RESTART** VLAN_MEMBER PORT|Ethernet90 INTERFACE|Ethernet116|xxxx:f4x7:470:a750::2/126 ACL_RULE|NO-NSW-PACL-V6|Rule_500
STATE_DB	6	Operational state for objects in CONFIG_DB	PORT_TABLE|Ethernet106: VLAN_TABLE|Vlan567: VLAN_TABLE|Vlan234:



Redis DB: Key-Value data & Python libraries

```
"PORT|Ethernet4"

1) "alias"
2) "Eth2/1"
3) "lanes"
4) "69,70"
5) "description"
6) "xxG|switch-in-dc.nw|Ethernet112"
7) "fec"
8) "xx"

----------------------------------------------------

"ROUTE_TABLE:xxxx:f547:4551:21b::/64"

1) "ifname"
2)"Ethernet64,Ethernet66,Ethernet68,Ethernet7
0,Ethernet72,Ethernet74,Ethernet76,Ethernet78
3) "nexthop"
 4)xxxx:f547:40:4027::1,xxxx:f547:40:4067::1,xx
xx:f547:40:40a7::1,xxxx:f547:40:40e7::1,xxxx:f5
 47:40:4127::1,xxxx:f547:40:4167::1,xxxx:f547:4
                    0:41a7::1,xxxx:f547:40:41e7::1"
```

```
"ASIC_STATE:SAI_OBJECT_TYPE_ROUTE_
ENTRY:{\"dest\":\"xx.aaa.239.128/26\",\"switch
_id\":\"oid:0x21000000000000\",\"vr\":\"oid:0x3
000000000042\"}"

1)
"SAI_ROUTE_ENTRY_ATTR_NEXT_HOP_ID
"
2) "oid:0x500000000xxc2"
```

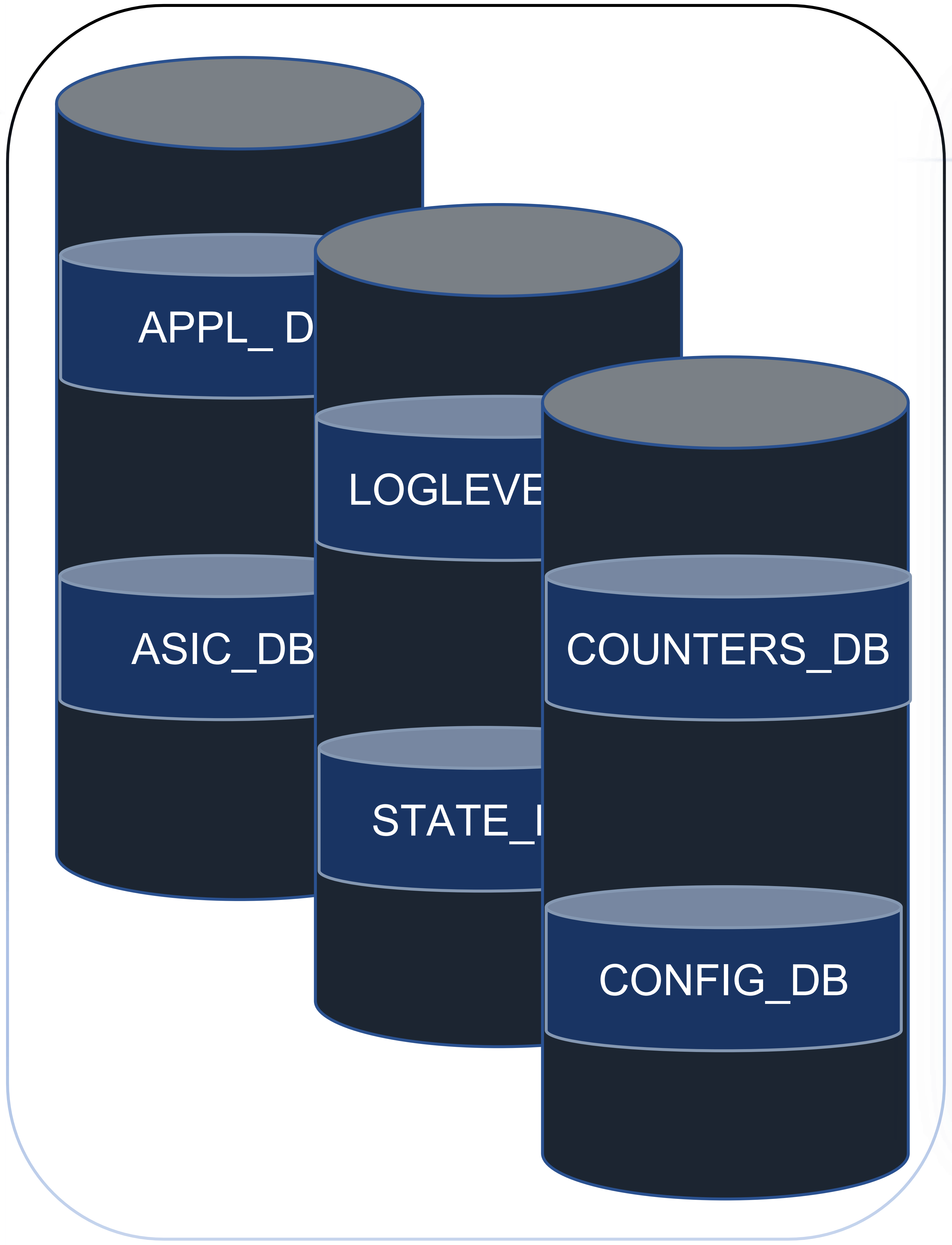


```
"hostname" : "127.0.0.1",
"port": 6379,
"unix_socket_path": "/var/run/redis/redis.sock"
```

```
sudo redis-cli -n  3 keys '*'

sudo redis-cli -n 3 hget
"neighsyncd:neighsyncd" "LOGLEVEL"
"NOTICE"

redis-cli -n 4 hget "ARP|arp2host" enable
```

https://github.com/Azure/sonic-buildimage/blob/master/dockers/docker-database/database_config.json

```
{
   "INSTANCES": {
      "redis":{
         "hostname" : "127.0.0.1",
         "port": 6379,
         "unix_socket_path": "/var/run/redis/redis.sock"
      },
      "redis1":{
         "hostname" : "127.0.0.1",
         "port": 6380,
         "unix_socket_path": "/var/run/redis/redis1.sock"
      }
      "redis2":{
         "hostname" : "127.0.0.1",
         "port": 6390,
         "unix_socket_path": "/var/run/redis/redis2.sock"
      }

}

  "DATABASES" : {
      "APPL_DB" : {
         "id" : 0,
         "instance" : "redis"
      },
      "ASIC_DB" : {
         "id" : 1,
         "instance" : "redis"
      },
      "COUNTERS_DB" : {
         "id" : 2,
         "instance" : "redis2"
      },
      "LOGLEVEL_DB" : {
         "id" : 3,
         "instance" : "redis1"
      },
      "CONFIG_DB" : {
         "id" : 4,
         "instance" : "redis2"
      },
```



Multi Redis Server

https://github.com/Azure/sonic-buildimage/blob/master/dockers/docker-database/docker-database-init.sh

```
mkdir -p /var/run/redis/sonic-db
if [ -f /etc/sonic/database_config.json ]; then
cp /etc/sonic/database_config.json /var/run/redis/sonic-db

else

cp /etc/default/sonic-db/database_config.json /var/run/redis/sonic-db

fi



mkdir -p /etc/supervisor/conf.d/

# generate all redis server supervisord configuration file

sonic-cfggen -j /var/run/redis/sonic-db/database_config.json -t
/usr/share/sonic/templates/supervisord.conf.j2 >
/etc/supervisor/conf.d/supervisord.conf



exec /usr/bin/supervisord
```

Redis DB: interact using Python libraries

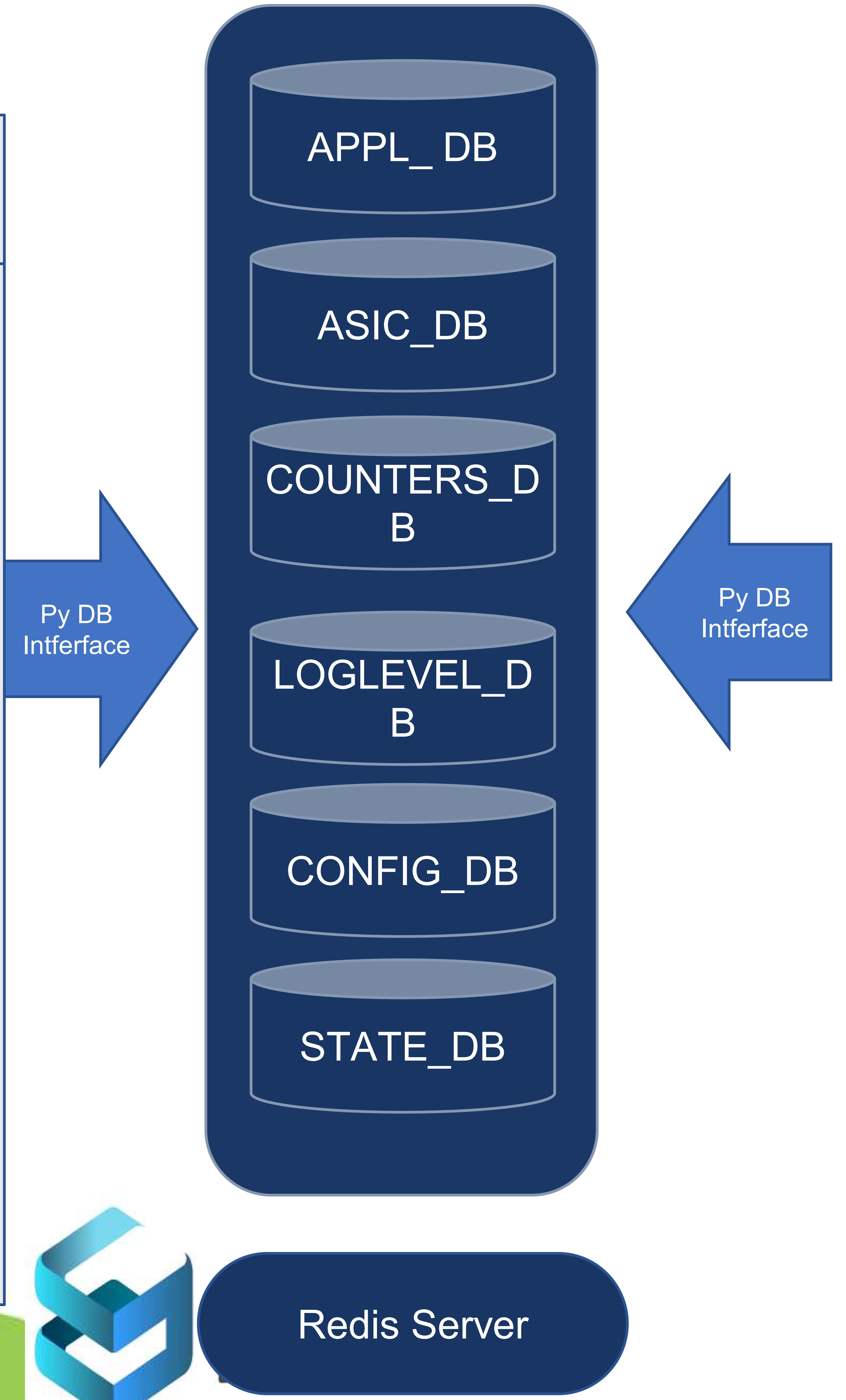
Main Classes:

https://github.com/Azure/sonic-py-swsssdk/blob/master/src/swsssdk/interface.py

```
-------------------------------------------------------------------------------
class DBInterface(object):
    REDIS_HOST = '127.0.0.1'
    REDIS_PORT = 6379
    REDIS_UNIX_SOCKET_PATH = "/var/run/redis/redis.sock"
......
    db_map = dict()
    def __init__(self, **kwargs):
    def connect(self, db_name, retry_on=True):
    def _onetime_connect(self, db_name):
    def _persistent_connect(self, db_name):
    def _subscribe_keyspace_notification(self, db_name):
    def get_redis_client(self, db_name):
    def publish(self, db_name, channel, message):
   def exists(self, db_name, key):
    def keys(self, db_name, pattern='*'):
    def get(self, db_name, _hash, key):
    def get_all(self, db_name, _hash):
    def set(self, db_name, _hash, key, val):
-------------------------------------------------------------------------------
```

https://github.com/Azure/sonic-py-swsssdk/blob/master/src/swsssdk/dbconnector.py

```
class SonicV2Connector(DBInterface):
class ConfigDBConnector(SonicV2Connector):
```



Example Code:

Example1: Fetch all keys from VLAN table from Config DB.

```
kwargs = {}
if redis_unix_socket_path:
    kwargs['unix_socket_path'] = redis_unix_socket_path

config_db = ConfigDBConnector(**kwargs)
config_db.connect(wait_for_init=False)

data = config_db.get_table('VLAN')
keys = data.keys()
```

-

Example2: Fetch set of key-value pair for a bvid from ASIC DB:

```
db = SonicV2Connector (**redis_kwargs)
db.connect('ASIC_DB')

vlan_obj = db.keys('ASIC_DB',
"ASIC_STATE:SAI_OBJECT_TYPE_VLAN:" + bvid)

vlan_entry = db.get_all('ASIC_DB', vlan_obj[0],
blocking=True)

vlan_id = vlan_entry[b"SAI_VLAN_ATTR_VLAN_ID"]

>>>>Sample Output:
 vlan_obj =
["ASIC_STATE:SAI_OBJECT_TYPE_VLAN:oid:0x26000
0000012a8"]

vlan_entry {
         "SAI_VLAN_ATTR_VLAN_ID": "555"
 } <<<<<<
```

Redis DB: interact with c++ libraries

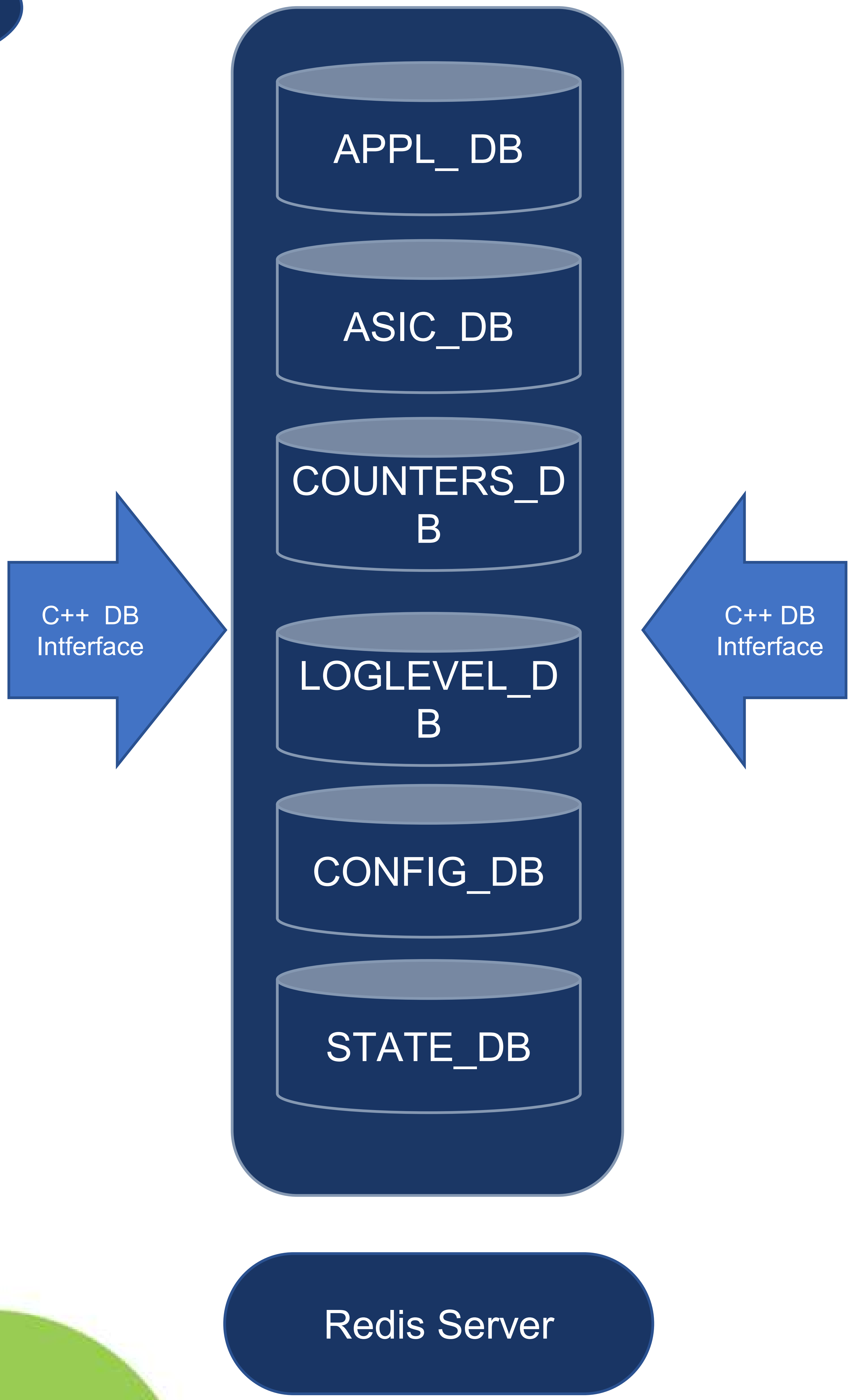
C++ Libraries:
https://github.com/Azure/sonic-swss-common/blob/master/common/dbconnector.cpp
https://github.com/Azure/sonic-swss-common/blob/master/common/table.cpp

```
----------------------------------------------------------------------
class DBConnector
{
public:
    static constexpr const char
*DEFAULT_UNIXSOCKET =
"/var/run/redis/redis.sock";

..........
}
----------------------------------------------------------------------
```

https://github.com/Azure/sonic-swss-common/ common/ *.h

(Global vars can be found here.)

```
#define CONFIG_DB       4
#define CFG_PORT_TABLE_NAME
"PORT"
#define
CONFIGDB_TABLE_NAME_SEPARATOR   "|"
```



Example Code:

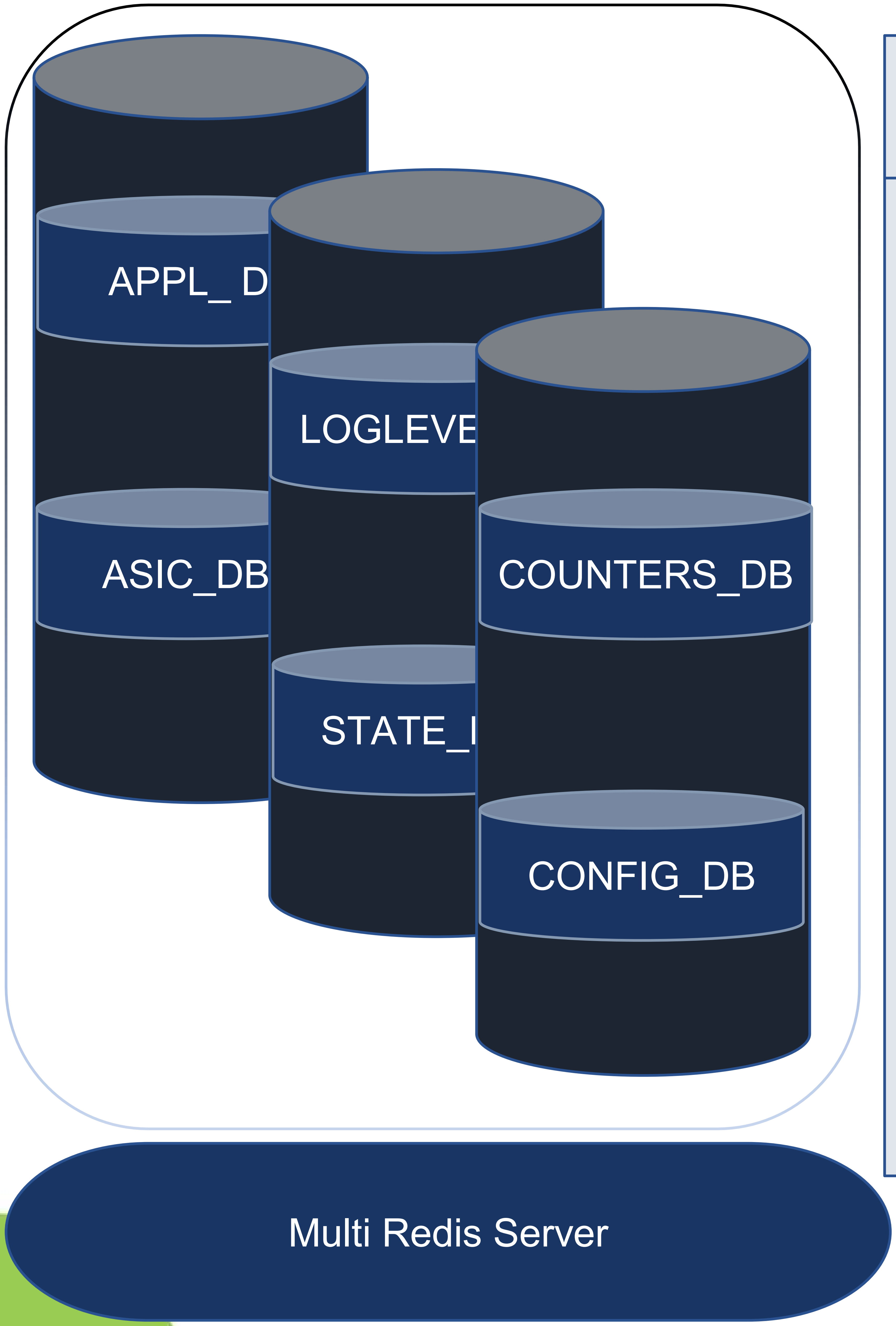
Access all <keys> and <key-value> pair from PORT_TABLE of CONFIG_DB using C++ libs:

```
----------
DBConnector cfgDb(CONFIG_DB,
DBConnector::DEFAULT_UNIXSOCKET, 0);

Table table(&cfgDb,
CFG_PORT_TABLE_NAME,
CONFIGDB_TABLE_NAME_SEPARATOR);

std::vector<FieldValueTuple> values;

std::vector<string> keys;table.getKeys(keys);

for ( auto &k : keys )
{
   table.get(k, ovalues);
   /----My Code ----/
}
```

Python Libraries:

https://github.com/Azure/sonic-py-swsssdk/blob/master/src/swsssdk/dbconnector.py

```
class SonicDBConfig(object):
    SONIC_DB_CONFIG_FILE = "/var/run/redis/sonic-db/database_config.json"
    _sonic_db_config_init = False
    _sonic_db_config = {}


class SonicV2Connector(DBInterface):

 def connect(self, db_name, retry_on=True):
        if self.use_unix_socket_path:
            self.redis_kwargs["unix_socket_path"] = self.get_db_socket(db_name)<<<<<<<</var/run/redis/redis[-n].sock

            self.redis_kwargs["host"] = None
            self.redis_kwargs["port"] = None
        else:
…

    def get_db_port(self, db_name):
        return SonicDBConfig.get_port(db_name)

    def get_dbid(self, db_name):
        return SonicDBConfig.get_dbid(db_name)


---------------------
```



C++ Libraries

https://github.com/Azure/sonic-swss-common/blob/master/common/dbconnector.h\cpp

```
-------------------------------------------------
  DBConnector::DBConnector(const string& dbName, unsigned int timeout, bool isTcpConn) :
    m_dbId(SonicDBConfig::getDbId(dbName))
{
    struct timeval tv = {0, (suseconds_t)timeout * 1000};

    if (timeout)
    {
        if (isTcpConn)
            m_conn = redisConnectWithTimeout(SonicDBConfig::getDbHostname(dbName).c_str(), SonicDBConfig::getDbPort(dbName), tv);

        else
            m_conn = redisConnectUnixWithTimeout(SonicDBConfig::getDbSock(dbName).c_str(), tv);<<<<<<<<<<<
    }
```

```
redis-cli -n 4 hget "ARP|arp2host" enable
```

swsssdk/src/script/**sonic-db-cli**

sonic-db-cli CONFIG_DB hget "ARP|arp2host" enable

Publisher:
----

https://github.com/Azure/sonic-swss-common/blob/master/common/producerstatetable.cpp

https://github.com/Azure/sonic-swss-common/blob/master/common/producertable.cpp

static constexpr **const char** *DEFAULT_UNIXSOCKET = **"/var/run/redis/redis[0-N].sock"**;

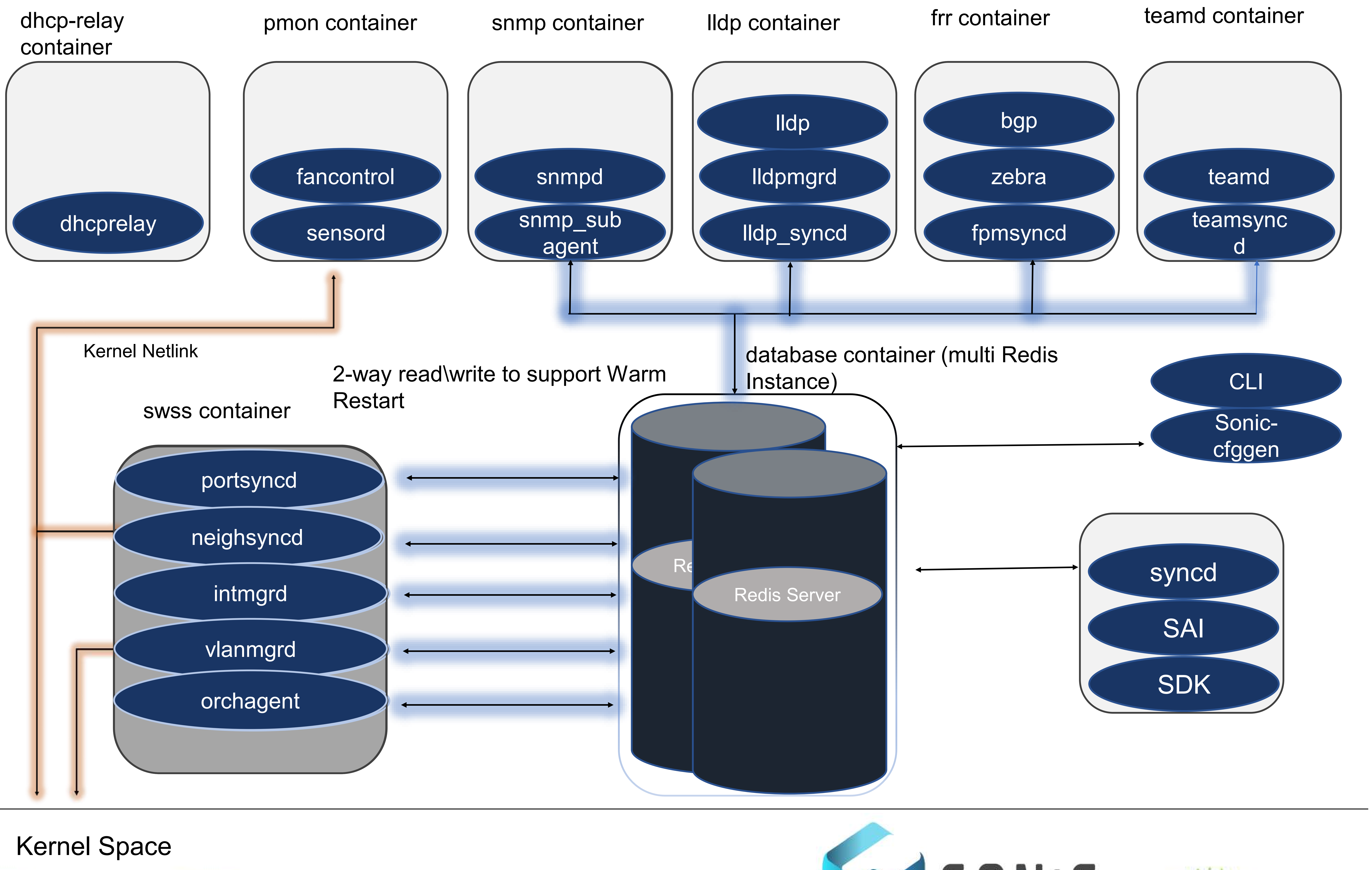
Subscriber:
----

https://github.com/Azure/sonic-swss-common/blob/master/common/subscriberstatetable.cpp

https://github.com/Azure/sonic-swss-common/blob/master/common/consumertablebase.cpp

https://github.com/Azure/sonic-swss-common/blob/master/common/consumerstatetable.cpp

----



WARM RELATED STATE\CONFIG ENTRIES IN REDIS DB:

```
WARM_RESTART_ENABLE_TABLE
;Stores system warm start and docker warm start enable/disable
configuration
;The configuration is persistent across warm reboot but not cold reboot.
;Status: work in progress
key                = WARM_RESTART_ENABLE_TABLE:name ; name is the
name of SONiC docker or "system" for global configuration.
enable             = "true" / "false"  ; Default value as false.
                             ; If "system" warm start knob is true, docker level
knob will be ignored.
                             ; If "system" warm start knob is false, docker level
knob takes effect.
```

```
WARM_RESTART
;Stores system warm start configuration
;Status: work in progress
key                = WARM_RESTART:name ; name is the name of SONiC
docker or "system" for global configuration.
neighsyncd_timer    = 1*4DIGIT
bgp_timer           = 1*4DIGIT
```

```
WARM_RESTART_TABLE
;Stores application and orchdameon warm start status
;Status: work in progress
key             = WARM_RESTART_TABLE|process_name
restore_count   = 1*10DIGIT
state           = "initialized" / "restored" / "reconciled"
```

WARM RESTART BASE CLASSES:

sonic-swss-common/common/warm_restart.cpp

```
void WarmStart::initialize(const std::string &app_name,
                           const std::string &docker_name,

bool WarmStart::checkWarmStart(const std::string &app_name,
                               const std::string &docker_name,
                               const bool incr_restore_cnt)

uint32_t WarmStart::getWarmStartTimer(const std::string &app_name,
                                      const std::string &docker_name)
```

https://github.com/Azure/sonic-swss/blob/master/warmrestart/warmRestartAssist.cpp

```
void AppRestartAssist::readTablesToMap()

void AppRestartAssist::insertToMap(string tableName, string key, vector<FieldValueTuple> fvVector, bool delete_key)

void AppRestartAssist::reconcile()
```

https://github.com/Azure/sonic-swss/blob/master/warmrestart/warmRestartHelper.cpp

```
bool WarmStartHelper::runRestoration()

void WarmStartHelper::reconcile(void)
```

https://github.com/Azure/sonic-swss-common/blob/master/common/producerstatetable.cpp

```
void ProducerStateTable::apply_temp_view()

void ProducerStateTable::apply_temp_view()
```

*_Syncd Processes: (Without WarmRestart)

lldpd
lldpmgrd
Lldp_syncd
LLDP
Bgpd
Zebra
FPMSyncd
FRR
Teamd
Teamsyncd
TeamD
PortSyncd
NeighSyncd
SWSS Docker
APPL_ DB
ASIC_DB
CONFIG_D B
STATE_DB
Kernel

Code Path:
https://github.com/Azure/sonic-swss/blob/master/neighsyncd
https://github.com/Azure/sonic-swss/blob/201803/ neighsyncd
(before warm restart feature)

```
 --------------------------------------------------------------------------
Class NeighSync: public NetMsg
{
…….
producerStateTable m_neighTable;
}
--------------------------------------------------------------------------
/* Main Function */
Main():
DBConnector db(APPL_DB, DBConnector::DEFAULT_UNIXSOCKET, 0
NeighSync sync(&db);
--------------------------------------------------------------------------
NeighSync::NeighSync(DBConnector *db) :
m_neighTable(db, APP_NEIGH_TABLE_NAME)
{}
--------------------------------------------------------------------------
void NeighSync::onMsg(int nlmsg_type, struct nl_object *obj)
….
nl_addr2str(rtnl_neigh_get_dst(neigh), ipStr, MAX_ADDR_SIZE);
….
nl_addr2str(rtnl_neigh_get_lladdr(neigh), macStr, MAX_ADDR_SIZE);
…..
FieldValueTuple f("family", family);
FieldValueTuple nh("neigh", macStr);
fvVector.push_back(nh);
fvVector.push_back(f);
m_neighTable.set(key, fvVector);
}
--------------------------------------------------------------------------
```

Code Path:
https://github.com/Azure/sonic-swss/blob/201803/fpmsyncd/fpmsyncd.cpp

```
--------------------------------------------------------------------------------------------------------
DBConnector db(APPL_DB, DBConnector::DEFAULT_UNIXSOCKET, 0);
RedisPipeline pipeline(&db);
RouteSync sync(&pipeline);<<<<<<<<<<[Class to Process the netlink from zebra]

NetDispatcher::getInstance().registerMessageHandler(RTM_NEWROUTE, &sync);
NetDispatcher::getInstance().registerMessageHandler(RTM_DELROUTE, &sync);

--------------------------------------------------------------------------------------------------------
```

https://github.com/Azure/sonic-swss/blob/201803/fpmsyncd/routesync.cpp

```
void RouteSync::onMsg(int nlmsg_type, struct nl_object *obj) {
…..
nl_addr2str(dip, destipprefix, MAX_ADDR_SIZE);

struct rtnl_nexthop *nexthop = rtnl_route_nexthop_n(route_obj, i);
…..
FieldValueTuple nh("nexthop", nexthops);
FieldValueTuple idx("ifname", ifnames);
fvVector.push_back(nh);
fvVector.push_back(idx);
m_routeTable.set(destipprefix, fvVector);
}
```

*_Syncd Processes: (With WarmRestart

lldpd
lldpmgrd
Lldp_syncd
LLDP
Bgpd
Zebra
FPMSyncd
FRR
Teamd
Teamsyncd
TeamD
PortSyncd
NeighSyncd
SWSS Docker
APPL_ DB
ASIC_DB
CONFIG_D B
STATE_DB
Kernel

Code Path:
https://github.com/Azure/sonic-swss/blob/master/neighsyncd/neighsyncd.cpp

```
 ---------------------------------------------------------------------------
          /*
           * If warmstart, read neighbor table to cache map.
           * Wait the kernel neighbor table restore to finish in case of
warmreboot.
           * Regular swss docker warmstart should have marked the restore
flag to true always.
           * Start reconcile timer once restore flag is set
           */

          if (sync.getRestartAssist()->isWarmStartInProgress())
<<<<<<<<<
          {

             sync.getRestartAssist()->readTablesToMap(); <<<<<<<<<

   ..........

   sync.getRestartAssist()->startReconcileTimer(s); <<<<<<<<<
s

   ............
   /*
           * If warmstart is in progress, we check the reconcile timer,
           * if timer expired, we stop the timer and start the reconcile proces
           */
          if (sync.getRestartAssist()->isWarmStartInProgress())
          {
               if (sync.getRestartAssist()->checkReconcileTimer(temps))
               {
                  sync.getRestartAssist()->stopReconcileTimer(s);

                  sync.getRestartAssist()->reconcile(); <<<<<<<<<
               }
```

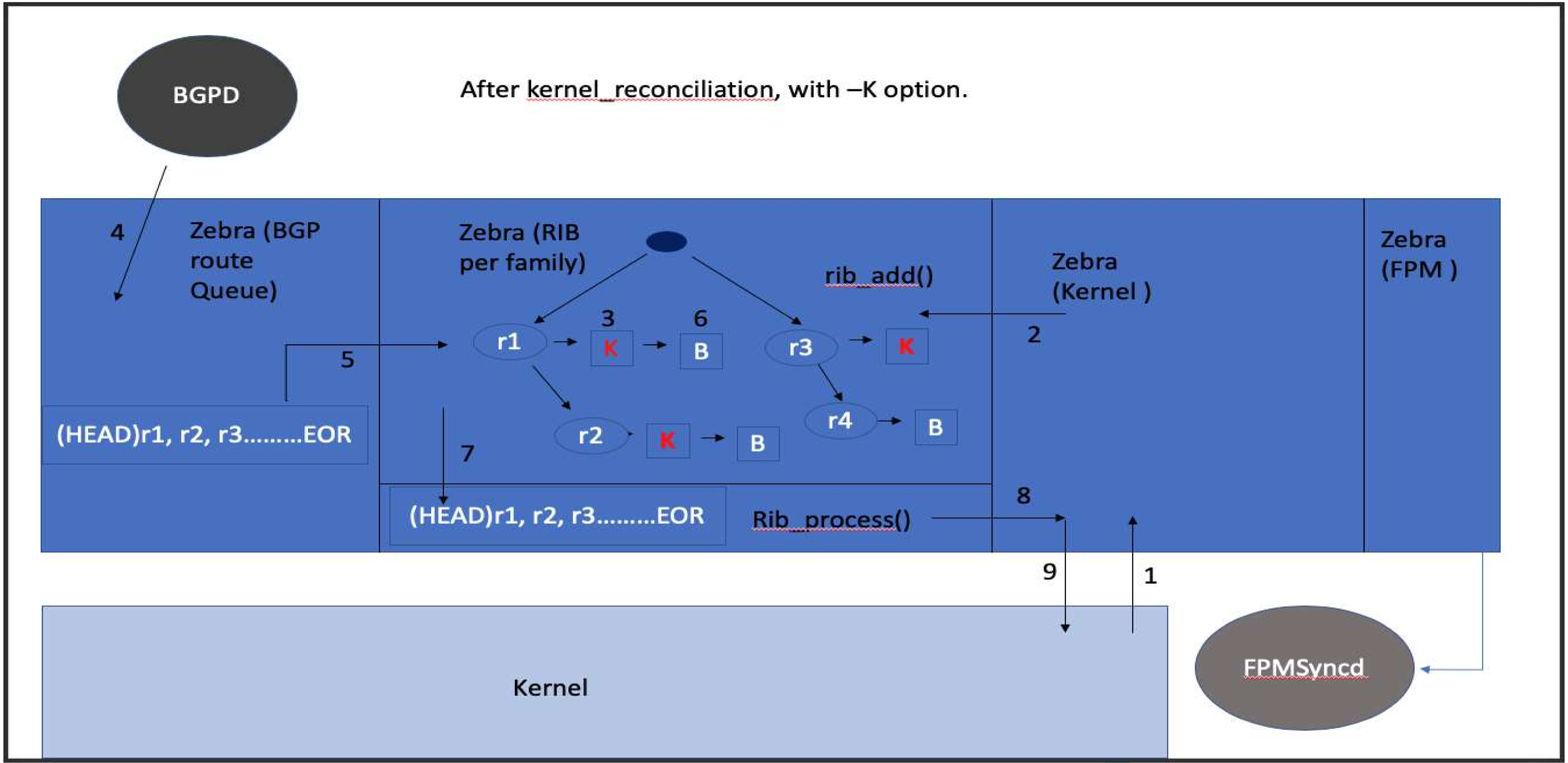
Code Path:
https://github.com/Azure/sonic-swss/blob/master/fpmsyncd/fpmsyncd.cpp ---

```
--------------------------------------------------------------------------------------------------------
----
 /* If warm-restart feature is enabled, execute 'restoration' logic */
    bool warmStartEnabled =
sync.m_warmStartHelper.checkAndStart();<<<<<

   /* Execute restoration instruction and kick off warm-restart timer */
    if (sync.m_warmStartHelper.runRestoration()) <<<<<<<<<

……………………
sync.m_warmStartHelper.reconcile(); <<<<<<<<<

--------------------------------------------------------------------------------------------------------
-------
```

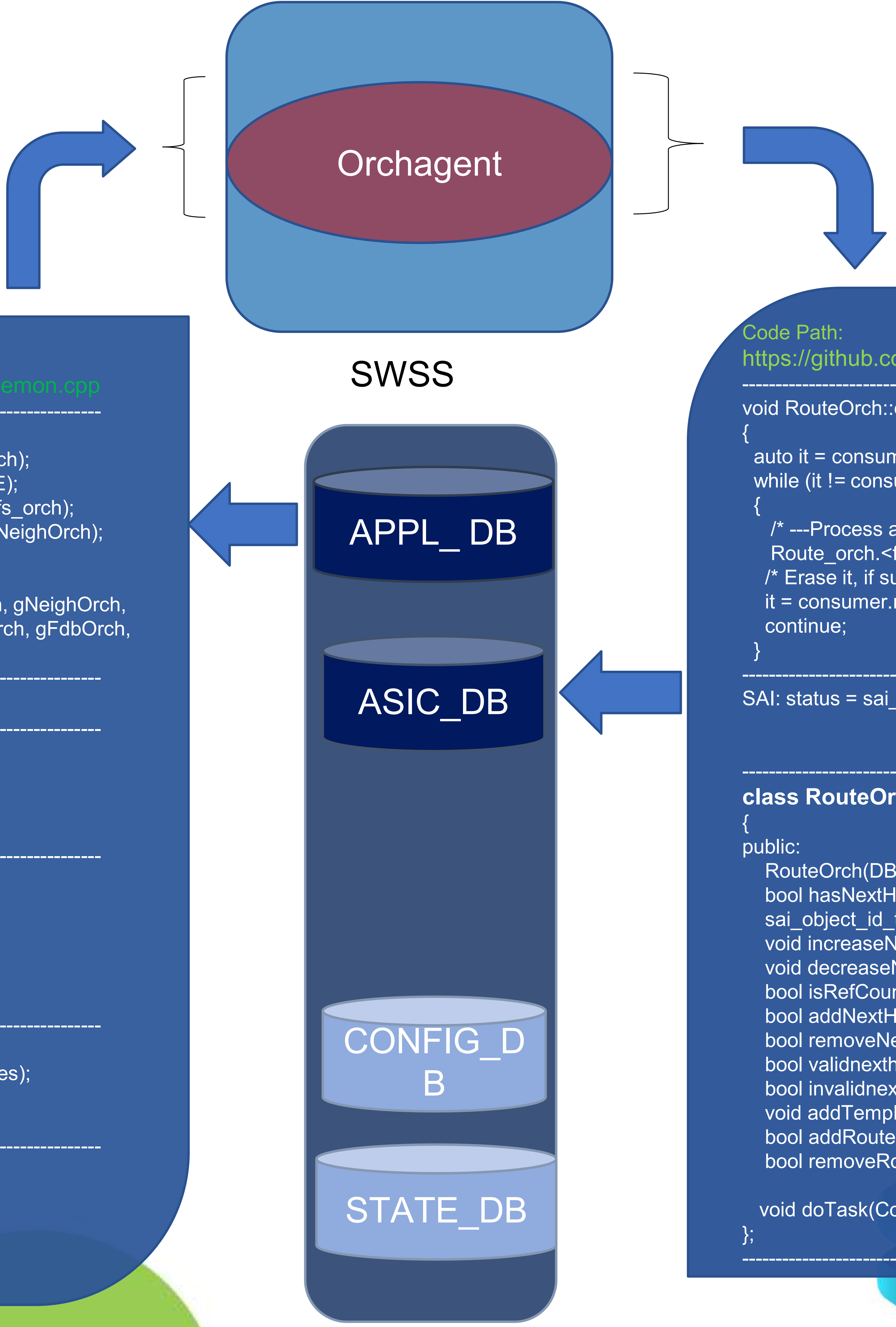
https://github.com/Azure/sonic-swss/blob/master/fpmsyncd/routesync.cpp

```
/*
 * Upon arrival of a delete msg we could either push the change right away,
 * or we could opt to defer it if we are going through a warm-reboot cycle.
 */
 bool warmRestartInProgress = m_warmStartHelper.inProgress();

/*
 * During routing-stack restarting scenarios route-updates will be
temporarily
 * put on hold by warm-reboot logic.
 */
m_warmStartHelper.insertRefreshMap(kfv); <<<<<<<<<
```

After kernel_reconciliation, with –K option.



Orchagent Processes:



Code Path: https://github.com/Azure/sonic-swss/blob/201803/orchagent/orchdaemon.cpp

```
gPortsOrch = new PortsOrch(m_applDb, ports_tables);
gFdbOrch = new FdbOrch(m_applDb, APP_FDB_TABLE_NAME, gPortsOrch);
IntfsOrch *intfs_orch = new IntfsOrch(m_applDb, APP_INTF_TABLE_NAME);
gNeighOrch = new NeighOrch(m_applDb, APP_NEIGH_TABLE_NAME, intfs_orch);
gRouteOrch = new RouteOrch(m_applDb, APP_ROUTE_TABLE_NAME, gNeighOrch);
```

***Orch Classes**

```
m_orchList = { switch_orch, gCrmOrch, gBufferOrch, gPortsOrch, intfs_orch, gNeighOrch,
gRouteOrch, copp_orch, tunnel_decap_orch, qos_orch, mirror_orch, gAclOrch, gFdbOrch,
vrf_orch };
```

class NeighOrch : public Orch, public Subject

Code Path: https://github.com/Azure/sonic-swss/blob/201803/orchagent/orch.cpp

```
class Consumer : public Executor {
…
    void execute();
    void drain();
    SyncMap m_toSync;
};
```

```
void Consumer::execute() {
                m_toSync[key] = KeyOpFieldsValuesTuple(key, op, existing_values);
….
}
```

```
void Consumer::drain()
{
    if (!m_toSync.empty())
        m_orch->doTask(*this);
}
```

Code Path: https://github.com/Azure/sonic-swss/blob/201803/orchagent/routeorch.cpp

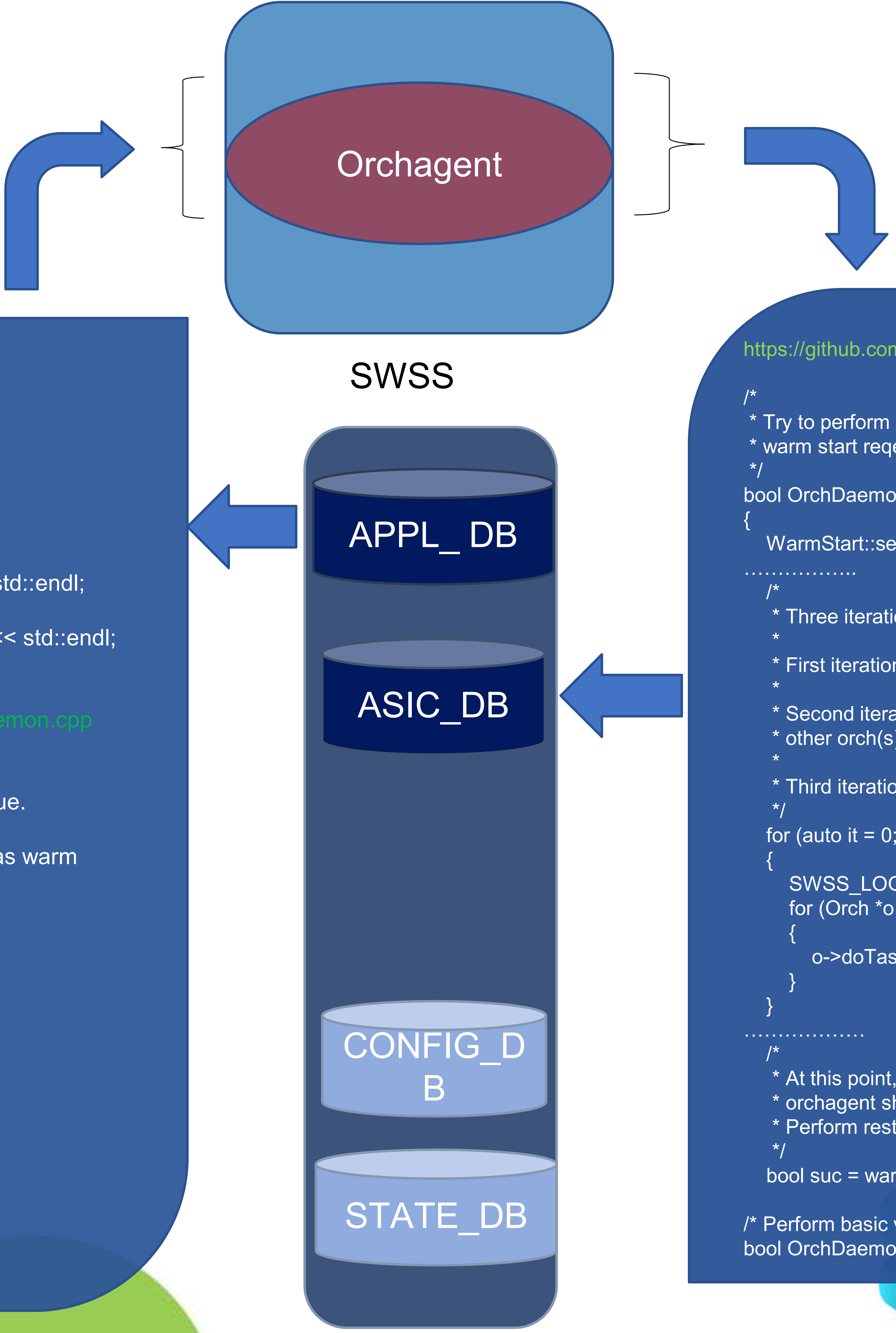
```
void RouteOrch::doTask(Consumer& consumer)
{
  auto it = consumer.m_toSync.begin();
  while (it != consumer.m_toSync.end())
  {
     /* ---Process as per the role */
     Route_orch.<func>() [SAI call()]
    /* Erase it, if success.
    it = consumer.m_toSync.erase(it);
    continue;
  }
```

```
SAI: status = sai_route_api->create_route_entry(&unicast_route_entry, 1, &attr);
```

class RouteOrch : public Orch, public Subject

```
{
public:
    RouteOrch(DBConnector *db, string tableName, NeighOrch *neighOrch);
    bool hasNextHopGroup(const IpAddresses&) const;
    sai_object_id_t getNextHopGroupId(const IpAddresses&);
    void increaseNextHopRefCount(IpAddresses);
    void decreaseNextHopRefCount(IpAddresses);
    bool isRefCounterZero(const IpAddresses&) const;
    bool addNextHopGroup(IpAddresses);
    bool removeNextHopGroup(IpAddresses);
    bool validnexthopinNextHopGroup(const IpAddress &);
    bool invalidnexthopinNextHopGroup(const IpAddress &);
    void addTempRoute(IpPrefix, IpAddresses);
    bool addRoute(IpPrefix, IpAddresses);
    bool removeRoute(IpPrefix);

   void doTask(Consumer& consumer);
};
```

Orchagent Processes: (Warm Restart)



https://github.com/Azure/sonic-swss/blob/master/orchagent/orchagent_restart_check.cpp

```
std::cout << "Usage: orchagent_restart_check [-s] " << std::endl;
std::cout << " -n --noFreeze" << std::endl;
std::cout << " Don't freeze orchagent even if check succeeded" << std::endl;
std::cout << " -s --skipPendingTaskCheck" << std::endl
std::cout << " Skip pending task dependency check for orchagent" << std::endl;
```

https://github.com/Azure/sonic-swss/blob/master/orchagent/orchdaemon.cpp

```
/*
 * Reply with "READY" notification if no pending tasks, and return true.
 * Ortherwise reply with "NOT_READY" notification and return false.
 * Further consideration is needed as to when orchagent is treated as warm
restart ready.
 * For now, no pending task should exist in any orch agent.
 */
bool OrchDaemon::warmRestartCheck() <<<<<<<<<<<<<
{
    std::vector<swss::FieldValueTuple> values;
    std::string op = "orchagent";
    std::string data = "READY";
    bool ret = true;

    vector<string> ts;
    getTaskToSync(ts);
```

https://github.com/Azure/sonic-swss/blob/master/orchagent/orchdaemon.cpp

```
/*
 * Try to perform orchagent state restore and dynamic states sync up if
 * warm start reqeust is detected.
 */
bool OrchDaemon::warmRestoreAndSyncUp() <<<<<<<<<<<<<
{
  WarmStart::setWarmStartState("orchagent", WarmStart::INITIALIZED);
………………
  /*
   * Three iterations are needed.
   *
   * First iteration: switchorch, Port init/hostif create part of portorch, buffers configuration
   *
   * Second iteratoin: port speed/mtu/fec_mode/pfc_asym/admin_status config,
   * other orch(s) which wait for port to become ready.
   *
   * Third iteration: Drain remaining data that are out of order.
   */
  for (auto it = 0; it < 3; it++)
  {
    SWSS_LOG_DEBUG("The current iteration is %d", it);
    for (Orch *o : m_orchList)
    {
      o->doTask();
    }
  }
………………
  /*
   * At this point, all the pre-existing data should have been processed properly, and
   * orchagent should be in exact same state of pre-shutdown.
   * Perform restore validation as needed.
   */
  bool suc = warmRestoreValidation();

/* Perform basic validation after start restore for warm start */
bool OrchDaemon::warmRestoreValidation() <<<<<<<<<<<<
```

[Access Control List]
aclorch.cpp
aclorch.h

[Control Plane Policy]
copporch.cpp
copporch.h

[Forwarding DataBase]
fdborch.cpp
fdborch.h

[Interfacse]
intfsorch.cpp
intfsorch.h

[Mirror]
mirrororch.cpp
mirrororch.h

[Neighbor]
neighorch.cpp
neighorch.h

[Mirror]
mirrororch.cpp
mirrororch.h

[Neighbor]
neighorch.cpp
neighorch.h

[Base Orch Class]
orch.cpp
orch.h

[Priority Flow Control]
pfcwdorch.cpp
pfcwdorch.h

[Port]
portsorch.cpp
portsorch.h

[Route]
routeorch.cpp
routeorch.h

[Tunnel Decap]
tunneldecaporch.cpp
tunneldecaporch.h

[Virtual Routing and Forwarding]
vrforch.cpp
vrforch.h

https://github.com/Azure/sonic-swss/blob/master/orchagent

List of open source code repo used in Sonic:

FRR & Zebra: https://github.com/FRRouting/frr

LLDP: https://github.com/vincentbernat/lldpd.git

LLDPMGRD: https://github.com/Azure/sonic-buildimage/blob/master/dockers/docker-lldp-sv2/lldpmgrd

SNMP: https://sourceforge.net/projects/net-snmp/files/net-snmp/5.7.3/

Sonic_snmp agent: https://github.com/Azure/sonic-snmpagent/tree/master/src

Teamd: https://salsa.debian.org/debian/libteam, https://github.com/jpirko/libteam.git

Dhcp_Relay: https://salsa.debian.org/berni/isc-dhcp.git

CLI: https://github.com/Azure/sonic-utilities

Linux Kernel: https://github.com/torvalds/linux

Open for All.

OCP GLOBAL SUMMIT

MARCH 4 & 5, 2020 | SAN JOSE, CA