Procedure to Restore CRD from Bad State in CPS

Contents

Introduction

This document describes the procedure to restore Cisco Policy Suite (CPS) Custom Reference Data (CRD) table from BAD state.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Linux
- CPS
- MongoDB

Cisco recommends that you must have privilege access:

- Root access to CPS CLI
- "qns-svn" user access to CPS GUIs (Policy builder and CPS Central)

Components used

The information in this document is based on these software and hardware versions:

- CPS 20.2
- MongoDB v3.6.17
- UCS-B

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is

live, ensure that you understand the potential impact of any command.

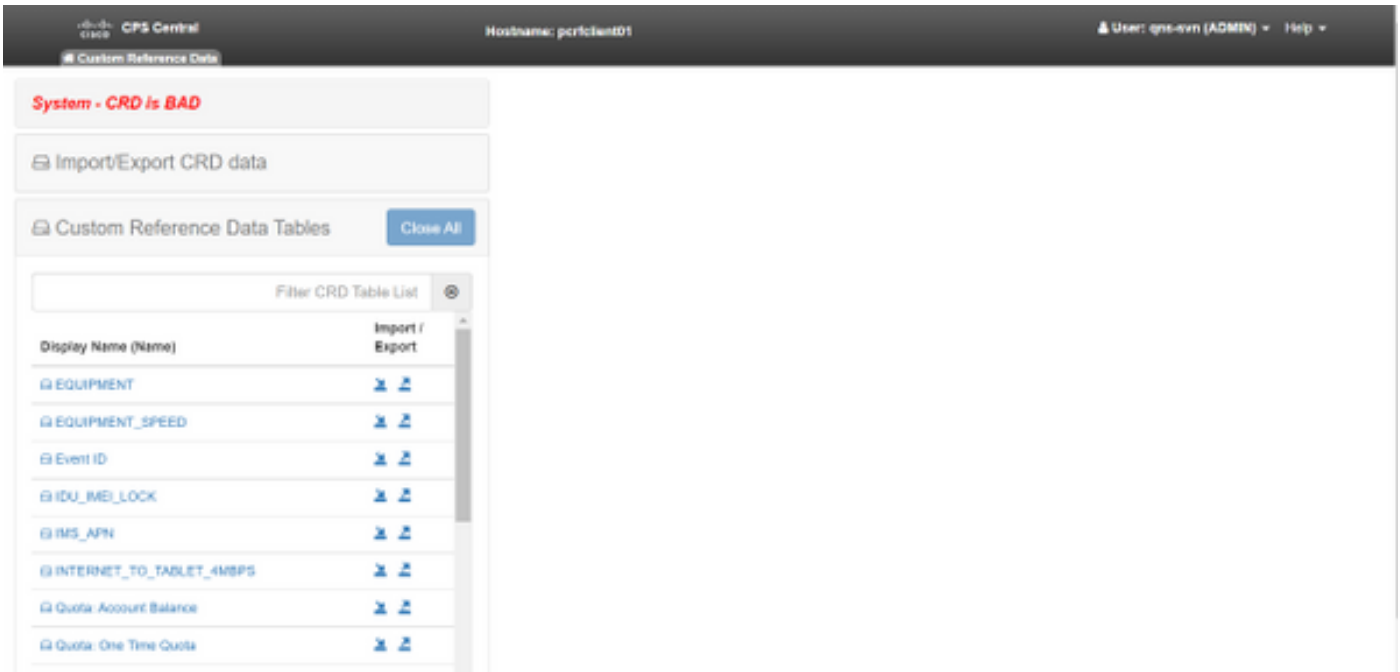
Background Information

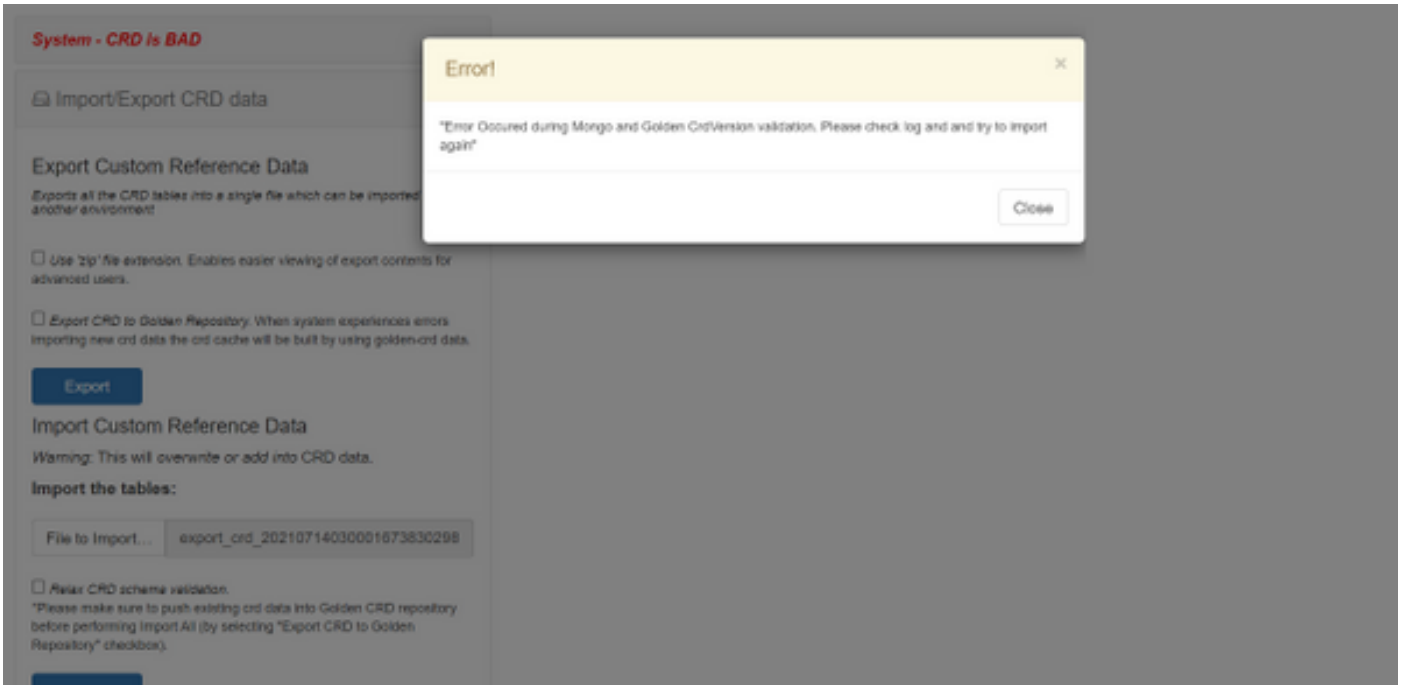
In CPS, CRD table is used to store custom policy configuration information which gets published from Policy Builder and associated with CRD DB which is present in MongoDB instance hosted on sessionmgr. Export and import operations are performed in the CRD table through CPS Central GUI in order to manipulate CRD table data.

Problem

If there is any kind of error when you perform import all operations, then CPS stops the process, sets the system in BAD state and blocks CRD APIs execution. CPS sends an error response to the client that states the system is in BAD state. If the system is in BAD state and you restarts Quantum Network Suite (QNS)/User Data Channel (UDC) server then CRD cache is built by the use of golden-crd data. If system BAD state is FALSE, then CRD cache is built with MongoDB.

Here are CPS Central Error images for reference.





If CRD system is BAD, then:

1. CRD manipulation is blocked. You can only view the data.
2. CRD APIs, except _import_all, _list, _query, are blocked.
3. QNS restart picks up CRD data from the golden-crd location.
4. A restart of the QNS/UDC neither fixes system BAD state nor call drops, it only builds CRD cache from golden-crd.
5. CRD cache built with golden-crd data. If the system BAD state is FALSE then crd cache built with MongoDB.

Here are associated messages in CPS qns.log:

```
qns02 qns02 2021-07-29 11:16:50,820 [pool-50847-thread-1]
INFO c.b.c.i.e.ApplicationInterceptor - System -
CRD is in bad state. All CRD APIs (except import all, list and query),
are blocked and user is not allowed to use.
Please verify your crd schema/crd data and try again!
qns02 qns02 2021-07-28 11:33:59,788 [pool-50847-thread-1]
WARN c.b.c.i.CustomerReferenceDataManager -
System is in BAD state. Data will be fetched from svn golden-crd repository.
qns01 qns01 2021-07-28 11:55:24,256 [pool-50847-thread-1]
WARN c.b.c.i.e.ApplicationInterceptor - ApplicationInterceptor: Is system bad: true
```

Procedure to Restore CRD from BAD State

Approach 1.

In order to clear the system state, you need to import valid and correct CRD schema from Policy Builder that involves the import of valid CRD data from CPS Central, if import all is successful then it clears the system state and all CRD APIs and operations are unblocked.

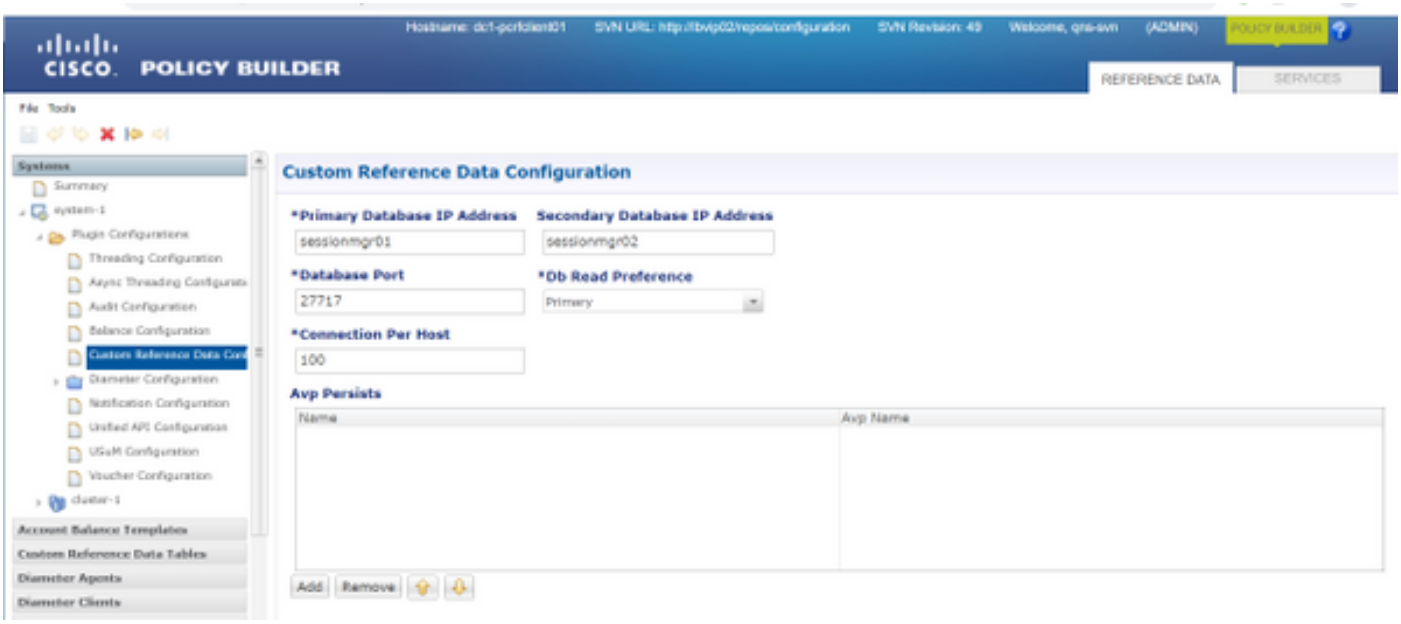
Provided here are the detailed steps:

Step 1. Run this command to Backup CRD Database.

```
Command template:
#mongodump --host <session_manager> --port <cust_ref_data_port>
--db cust_ref_data -o cust_ref_data_backup

Sample command:
#mongodump --host sessionmgr01 --port 27717 --db cust_ref_data -o cust_ref_data_backup
```

> **Note**: For CRD DB host and port, refer to Custom Reference Data Configuration in PB as shown in this image.



Step 2. Drop the CRD table (entire DB) with the use of this procedure.

Step 2.1. Log in to the mongo instance where CRD DB is present.

```
Command template:
#mongo --host <sessionmgrXX> --port <cust_ref_data_port>

Sample command:
#mongo --host sessionmgr01 --port 27717
```

Step 2.2. Run this command in order to display all the DBs present in the mongo instance.

```
set01:PRIMARY> show dbs
admin 0.031GB
config 0.031GB
cust_ref_data 0.125GB
local 5.029GB
session_cache 0.031GB
sk_cache 0.031GB
set01:PRIMARY>
```

Step 2.3. Run this command to Switch to CRD DB.

```
set01:PRIMARY> use cust_ref_data
```

```
switched to db cust_ref_data
set01:PRIMARY
```

Step 2.4. Run this command to Drop CRD DB.

```
set01:PRIMARY> db.dropDatabase()
{
"dropped" : "cust_ref_data",
"ok" : 1,
"operationTime" : Timestamp(1631074286, 13),
"$clusterTime" : {
"clusterTime" : Timestamp(1631074286, 13),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}}}
set01:PRIMARY>
```

Step 3. Verify that there is no db with name cust_ref_data that exists with the command **show dbs**.

```
set01:PRIMARY> show dbs
admin 0.031GB
config 0.031GB
local 5.029GB
session_cache 0.031GB
sk_cache 0.031GB
set01:PRIMARY>
```

Step 4. Log in to Policy builder with "qns-svn" user and publish valid CRD schema.

Step 5. Restart the qns process on all the nodes with **restartall.sh** from Cluster Manager.

Step 6. Verify that diagnostics is fine and there is no entry in CRD table. There must be only schema present in the CRD tables ie..without any data.

Step 7. Log in to CPS Central with "qns-svn" user and Import valid CRD data.

Step 8. Verify that, import all returns successful message and "system - CRD is BAD" error message not shown in CPS Central.

Step 9. Verify that, all CRD APIs are now unblocked, you can manipulate CRD data now.

If the first approach didn't work, then go for the second approach.

Approach 2.

Step 1. Identify the host and port in which ADMIN DB Mongo instance is hosted with the command **diagnostics.sh --get_r**.

```
[root@installer ~]# diagnostics.sh --get_r
CPS Diagnostics HA Multi-Node Environment
--------------------------
Checking replica sets...
|-----------------------------------------------------------------------------------------------
----------------------------------------|
| Mongo:v3.6.17 MONGODB REPLICA-SETS STATUS INFORMATION Date : 2021-09-14 02:56:23 |
|-----------------------------------------------------------------------------------------------
----------------------------------------|
| SET NAME - PORT : IP ADDRESS - REPLICA STATE - HOST NAME - HEALTH - LAST SYNC - PRIORITY |
|-----------------------------------------------------------------------------------------------
----------------------------------------|
| ADMIN:set06 |
| Status via arbitervip:27721 sessionmgr01:27721 sessionmgr02:27721 |
| Member-1 - 27721 : - PRIMARY - sessionmgr01 - ON-LINE - -------- - 3 |
| Member-2 - 27721 : - SECONDARY - sessionmgr02 - ON-LINE - 1 sec - 2 |
| Member-3 - 27721 : 192.168.10.146 - ARBITER - arbitervip - ON-LINE - -------- - 0 |
|-----------------------------------------------------------------------------------------------
----------------------------------------|
```

Step 2. Log in to the mongo instance where ADMIN DB is present.

```
Command template:
#mongo --host <sessionmgrXX> --port <Admin_DB__port>

Sample Command:
#mongo --host sessionmgr01 --port 27721
```

Step 3. Run this command to display all the DBs present in the mongo instance.

```
set06:PRIMARY> show dbs
admin 0.078GB
config 0.078GB
diameter 0.078GB
keystore 0.078GB
local 4.076GB
policy_trace 2.078GB
queueing 0.078GB
scheduler 0.078GB
sharding 0.078GB
set06:PRIMARY>
```

Step 4. Run this command to switch to ADMIN DB.

```
set06:PRIMARY> use admin
switched to db admin
set06:PRIMARY>
```

Step 5. Run this command to display all the tables present in ADMIN DB.

```
set06:PRIMARY> show tables
state
system.indexes
system.keys
system.version
set06:PRIMARY>
```

Step 6. Run this command to check the current state of the system.

```
set06:PRIMARY> db.state.find()
```

```
{ "_id" : "state", "isSystemBad" : true, "lastUpdatedDate" : ISODate("2021-08-11T15:01:13.313Z")
}
set06:PRIMARY>
```

Here, you can see that "**isSystemBad**" : **true**. Hence, you must update this field to "**false**" in-order to clear CRD BAD state, with the command provided in the next step.

Step 7. Update the field "isSystemBAD" with the command **db.state.updateOne({_id:"state"},{$set:{isSystemBad:false}})**.

```
set06:PRIMARY> db.state.updateOne({_id:"state"},{$set:{isSystemBad:false}})
{ "acknowledged" : true, "matchedCount" : 0, "modifiedCount" : 0 }
set06:PRIMARY>
```

Step 8. Run the command **db.state.find()** in order to check if **isSystemBad** field value has changed to **false.**

```
set06:PRIMARY> db.state.find()
{ "_id" : "state", "isSystemBad" : false, "lastUpdatedDate" : ISODate("2021-08-
11T15:01:13.313Z") }
set06:PRIMARY>
```

Step 9. Verify that all CRD APIs are now unblocked, you can manipulate CRD data now.