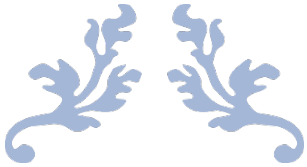

CAN - MIGRATION

Migration from Version 4.0 to 5.0

OCTOBER 13, 2020

AVANSEUS TECHNOLOGIES PVT. LTD.

Table of Contents

1. Objective

This document gives step by step procedure to upgrade the CAN 4.0 environment to CAN 5.0.

Note: Please take the dB backup before doing the following changes.

2. Update the database by adding new entries, removing or updating the existing entries to the existing collections

This section covers the following updates:

A. Adding new fields to **Cause** and **OfficeCode** collection and update its references in **Alarm**, **AlarmGUI** and **PredictedFault**,
B. User can have separate mail groups for the success and failure notifications for **Monitoring**. Multiple mail groups can be added.
C. New field added to **PredictionAssignmentPolicy** collection.
D. Remove the import statement related to MongoPersistenceManager in template collections
E. Addition of 'repetitiveAlarms' and 'isRepetitive' fields to the **PredictedFault** JSON for VBI usage.
F. New field added to EmployeeAccess collection and modified existing 'status' and 'expiryDate' field.
G. JSON structure changes related to PredictionKeyFilter collection and addition of new collection named PredictionFilterRule for Filter Configuration.

2.1 Cause related changes

Add domain, networkType, causeCategory, serviceAffecting & priority filed to the Cause collection. At the beginning, add the pre-defined random values to Cause collection. If required, please change the domain and network type array values as per the customer data.

Note: Domain can have multiple network types and network type should be unique across all the domains.

Run the MongoDB script (shown below) to apply the above mentioned changes:

```
/*remove the cause category field from Cause collection*/
db.Cause.updateMany({},{$unset:{"causeCategory":""}});


/*To add random domain, networkType, serviceAffecting, priority, cause category in cause table*/
function getArrayRandomElement (arr) {
  if (arr && arr.length) {
    return arr[Math.floor(Math.random() * arr.length)];
  }
}

var domainArray = ['CORE', 'TRANSPORT', 'ACCESS'];
var coreNetworkTypeArray = ['CS-CORE','PS-CORE'];
var transportNetworkTypeArray = ["CEN", "MPLS"];
var accessNetworkTypeArray = ['2G','3G','4G'];
var networkTypeArray;
var causeCategoryArray = ["INFRA","HARDWARE"];//Don't change this value
var booleanArray = [true,false];

db.Cause.find().forEach(function(doc){
  var cause = {};
  cause.domain = getArrayRandomElement(domainArray);
  if(cause.domain === "CORE")
    networkTypeArray = coreNetworkTypeArray;
  else if(cause.domain === "TRANSPORT")
    networkTypeArray = transportNetworkTypeArray;
```

```
  else if(cause.domain === "ACCESS")
    networkTypeArray = accessNetworkTypeArray;
  cause.networkType = getArrayRandomElement(networkTypeArray);
  cause.causeCategory = getArrayRandomElement(causeCategoryArray);
  cause.serviceAffecting = getArrayRandomElement(booleanArray);
  cause.priority = getArrayRandomElement(booleanArray);
  db.Cause.update({"_id":doc._id},
    {$set:
        { "causeCategory":cause.causeCategory,
          "domain":cause.domain,
          "networkType":cause.networkType,
          "serviceAffecting":cause.serviceAffecting,
          "priority":cause.priority,
        }
    });
});
```

After running the above script, open the CAN5.0 UI in your browser.
For Cause related changes, go to settings module > Cause management. Download the existing Cause entries using file download option. For each cause in the file, select the appropriate values for domain, networkType, priority, service affecting and cause category. Domain experts input is required here.

i. Priority and service affecting value should be YES or NO.
ii. Domain value should be CORE or TRANSPORT or ACCESS or as per the customer data.
iii. NetworkType should be relevant to domain and it should be unique across all the domains.
iv. Cause Category must be either INFRA or HARDWARE

Note: All the values should be in UPPER CASE.

```
/*To add cause JSON in Alarm collection*/
/*=> Alarm Table*/
/*Add index for cause_id in Alarm collection before running this script*/
var count = db.Cause.find().count();
var i = 0;
var bulk = db.Alarm.initializeUnorderedBulkOp();
print("Total cause to update "+count);
db.Cause.find().forEach(function(doc){
   var cause = {};
   cause.name = doc.name;
   cause.domain = doc.domain;
   cause.networkType = doc.networkType;
   cause.causeCategory = doc.causeCategory;
   cause.serviceAffecting = doc.serviceAffecting;
   cause.priority = doc.priority;
   var cause_id = doc._id.valueOf();
   bulk.find({"cause_id":cause_id}).update({"$set":{"cause":cause}});
   count--;
   i++;
   if(i == 50){
     bulk.execute();
     print("Bulk operations executed for 50 causes");
     print("Remaining cause "+ count);
     bulk = db.Alarm.initializeUnorderedBulkOp();
     i = 0;
   }
});
if(i> 0){
  bulk.execute();
  print("Bulk operations executed for last " + i +" causes");
}


/*=> Alarm GUI collection*/
/*Add index for cause_id in AlarmGUI collection before running this script*/
```

```
var count = db.Cause.find().count();
var i = 0;
var bulk = db.AlarmGUI.initializeUnorderedBulkOp();
print("Total cause to update "+count);
db.Cause.find().forEach(function(doc){
  var cause = {};
  cause.name = doc.name;
  cause.domain = doc.domain;
  cause.networkType = doc.networkType;
  cause.causeCategory = doc.causeCategory;
  cause.serviceAffecting = doc.serviceAffecting;
  cause.priority = doc.priority;
  var cause_id = doc._id.valueOf();
  bulk.find({"cause_id":cause_id}).update({
  "$set":{
        "cause":cause
  }});
  count--;
  i++;
  if(i == 50){
    bulk.execute();
    print("Bulk operations executed for 50 causes");
    print("Remaining cause "+ count--);
    bulk = db.AlarmGUI.initializeUnorderedBulkOp();
    i = 0;
  }
});
if(i> 0){
 bulk.execute();
 print("Bulk operations executed for last " + i +" causes")
}

/*To add cause new fields in PredictedFault collection*/
/*Add index for "cause.name" in PredictedFault collection before running this script*/
var count = db.Cause.find().count();
var i = 0;
var bulk = db.PredictedFault.initializeUnorderedBulkOp();
print("Total cause to update "+count);
db.Cause.find().forEach(function(doc){
  var cause = {};
  cause.domain = doc.domain;
  cause.networkType = doc.networkType;
  cause.causeCategory = doc.causeCategory;
  cause.serviceAffecting = doc.serviceAffecting;
  cause.priority = doc.priority;
  var cause_name = doc.name;
  bulk.find({"cause.name":cause_name}).update(
  {"$set":{
        "cause.domain":cause.domain,
        "cause.networkType":cause.networkType,
              "cause.causeCategory":cause.causeCategory,
              "cause.serviceAffecting":cause.serviceAffecting,
              "cause.priority":cause.priority
  }});
  count--
  i++;
  if(i == 50){
    bulk.execute();
    print("Bulk operations executed for 50 causes");
    print("Remaning cause "+ count);
    bulk = db.PredictedFault.initializeUnorderedBulkOp();
    i = 0;
  }
});
if(i> 0){
 bulk.execute();
 print("Bulk operations executed for last " + i +" causes")
```

```
}
```

2.2 Changes related to Office Code.

Office Code priority has been newly added to the office code json. Add pre-defined random office code priorities to the Office code json at the beginning.

Run the MongoDB script (shown below) to apply the above mentioned changes:

```
/*To add random officeCodePriority in officeCode collection*/
function getArrayRandomElement (arr) {
  if (arr && arr.length) {
    return arr[Math.floor(Math.random() * arr.length)];
  }
}
var officeCodePriorityArray = ['CRITICAL', 'MAJOR', 'MINOR'];
db.OfficeCode.find().forEach(function(doc){
  var officeCode = {};
  officeCode.officeCodePriority = getArrayRandomElement(officeCodePriorityArray);
  db.OfficeCode.update({"_id":doc._id},
    {$set:
        {"officeCodePriority":officeCode.officeCodePriority,
        }
    });
});
```

For Office Code related changes, go to settings module in the CAN 5.0 UI, Settings > Priority Management. Download the existing Office Code entries using the file download option. For each office code, select the appropriate value for Site Priority. It can have values either CRITICAL, MAJOR or MINOR.
Note: All the Site Priority Values should be in UPPER CASE.
Once the above procedure is completed, run the below scripts.

```
/*To add officeCode JSON in Alarm JSON*/
/*=> Alarm collection*/
/*Add index for officeCode_id in Alarm collection before running this script*/
var count = db.OfficeCode.find().count();
var i = 0;
var bulk = db.Alarm.initializeUnorderedBulkOp();
print("Total office code to update "+count);
db.OfficeCode.find().forEach(function(doc){
  var officeCode = {};
  officeCode.name = doc.name;
  officeCode.officeCodePriority = doc.officeCodePriority;
  var officeCode_id = doc._id.valueOf();
  bulk.find({"officeCode_id":officeCode_id}).update({
  "$set":{
        "officeCode":officeCode
  }});
  count--;
  i++;
  if(i == 50){
    bulk.execute();
    print("Bulk operations executed for 50 office codes");
    print("Remaining office code "+ count);
    bulk = db.Alarm.initializeUnorderedBulkOp();
    i = 0;
  }
});
if(i> 0){
  bulk.execute();
  print("Bulk operations executed for last " + i +" office codes")
}
```

```
/*=> Alarm GUI collection*/
/*Add index for officeCode_id in AlarmGUI collection before running this script*/
var count = db.OfficeCode.find().count();
var i = 0;
var bulk = db.AlarmGUI.initializeUnorderedBulkOp();
print("Total office code to update "+count);
db.OfficeCode.find().forEach(function(doc){
  var officeCode = {};
  officeCode.name = doc.name;
  officeCode.officeCodePriority = doc.officeCodePriority;
  var officeCode_id = doc._id.valueOf();
  bulk.find({"officeCode_id":officeCode_id}).update({
  "$set":{
        "officeCode":officeCode
  }});
  count--;
  i++;
  if(i == 50){
    bulk.execute();
    print("Bulk operations executed for 50 office codes");
    print("Remaining office code "+ count);
    bulk = db.AlarmGUI.initializeUnorderedBulkOp();
    i = 0;
  }
});
if(i> 0){
 bulk.execute();
 print("Bulk operations executed for last " + i +" office codes")
}
```

```
/*To add officeCode JSON in Predicted Fault collection*/
/*Add index for "officeCode.name" in PredictedFault collection before running this script*/
var count = db.OfficeCode.find().count();
var i = 0;
var bulk = db.PredictedFault.initializeUnorderedBulkOp();
print("Total office code to update "+count);
db.OfficeCode.find().forEach(function(doc){
  var officeCode = {};
  officeCode.name = doc.name;
  officeCode.officeCodePriority = doc.officeCodePriority;
  var officeCode_id = doc._id.valueOf();
  bulk.find({"officeCode.name":officeCode.name}).update({
  "$set":{
        "officeCode.officeCodePriority":officeCode.officeCodePriority
  }});
  count--;
  i++;
  if(i == 50){
    bulk.execute();
    print("Bulk operations executed for 50 office codes");
    print("Remaining office code "+ count);
    bulk = db.PredictedFault.initializeUnorderedBulkOp();
    i = 0;
  }
});
if(i> 0){
 bulk.execute();
 print("Bulk operations executed for last " + i +" office codes");
}
```

2.3 Changes related to Monitoring.

Update the **AlertEmailConfiguration** collection as mentioned below. For every document in **AlertEmailConfiguration**, add the **successMailGroup_id** and **failureMailGroup_id** fields with the below format. Both can have multiple mail groups.

```
"successMailGroup_id" : [
        "<%mail_group_id_1%>",
        "<%mail_group_id_2%>",
        .
        .
        "<%mail_group_id_n%>"
],
"failureMailGroup_id" : [
        "<%mail_group_id_1%>",
        "<%mail_group_id_2%>",
        .
        .
        "<%mail_group_id_n%>"
]
```

For the existing entries, run the below Mongo script to update the JSON.

```
db.AlertEmailConfiguration.find({}).forEach(function(doc){
  var id = doc._id.valueOf();
  var mailGroupId = doc.mailGroup_id;
  var mailGroupIdArray = new Array();
  mailGroupIdArray.push(mailGroupId);
  db.AlertEmailConfiguration.updateOne({_id:ObjectId(id)},
  {$set:
      {
       "successMailGroup_id":mailGroupIdArray,
       "failureMailGroup_id":mailGroupIdArray
  },
  $unset:
   {
    mailGroup_id:""
   }
  });
});
```

After running the above script, go to Monitoring module in CAN 5.0 UI and configure the separate mail group for the success and failure notifications.

Sample JSON of **AlertEmailConfiguration** would look like below

```
{
    "_id" : ObjectId("5b14db8a4a49ea558f65bd77"),
    "key" : "Fault trace generation",
    "success_emailSubject" : "Fault trace generation on $date (Success)",
    "success_emailBody" : "Hi Team ,\n\nThe fault trace generation was completed successfully for the $date.
\nTotal count of fault trace is : $count. \n\nRegards,\nCAN Admin",
    "failure_emailSubject" : "Fault trace generation on $date (Failure)",
    "failure_emailBody" : "Hi Team ,\n\nThe fault trace generation was not completed for the $date. \nTotal
count of fault trace is : $count. \n\nRegards,\nCAN Admin",
    "isSuccessMailConfigured" : true,
    "isFailureMailConfigured" : true,
    "successMailGroup_id" : [
        "5e9e9cf9782258affbd2188d"
    ],
    "failureMailGroup_id" : [
        "5e9e9cf9782258affbd2188d"
    ]
```

```
}
```

2.4 Changes related to PredictionAssignmentPolicy.

In the PredictionAssignmentPolicy collection, add the **numberOfSavedNodes** field to each document and change the value type of numberOfNodes field from double to integer. Ignore if the numberOfNodes is having integer value.

Run the MongoDB script (shown below) to apply the above mentioned changes:

```
db.PredictionAssignmentPolicy.find({}).forEach(function(doc){
  var id = doc._id.valueOf();
  var numberOfNodes = doc.numberOfNodes;
  numberOfNodes = NumberInt(numberOfNodes);
  db.PredictionAssignmentPolicy.updateOne({_id:ObjectId(id)},
  {$set:
      {
       "numberOfNodes":numberOfNodes,
       "numberOfSavedNodes":numberOfNodes
  }});
});
```

2.5 Changes related to EventFileFormat, EventFileFormatTemplate, PostProcessor, PostProcessorTemplate, PreProcessor, PreProcessorTemplate, ExcelPageConfiguiration and ExcelPageConfigurationTemplate collection

i. Remove the "**import com.avanseus.database.mongo.MongoPersistenceManager;\n**" statement from **mappingInformation.mapping[i].generatedCode** field in EventFileFormat collection and from columnConfigurations in ExcelPageConfiguration collection

Run the below MongoDB script to apply the above mentioned changes:

```
//For EventFileFormat collection
db.EventFileFormat.find({}).forEach(function(doc){
  var mapping = doc.mappingInformation.mapping;
  var isGeneratedCodeFieldExists = false;
  for(var i=0;i<mapping.length;i++){
    if(mapping[i].generatedCode != undefined){
      mapping[i].generatedCode = mapping[i].generatedCode.replace(/import
com.avanseus.database.mongo.MongoPersistenceManager;\n/g, "");
      isGeneratedCodeFieldExists = true;
    }
  }
  if(isGeneratedCodeFieldExists){
     db.EventFileFormat.updateOne({_id:doc._id},
          {$set:
               {"mappingInformation.mapping":mapping}
     });
  }
});

//For ExcelPageConfiguration collection
db.ExcelPageConfiguration.find({}).forEach(function(doc){
  var columnConfigurations = doc.columnConfigurations;
  var isComplexFieldExists = false;
  for(var i=0;i< columnConfigurations.length;i++){
    if(columnConfigurations[i].complex != undefined){
      columnConfigurations[i].complex = columnConfigurations[i].complex.replace(/import
com.avanseus.database.mongo.MongoPersistenceManager;\n/g, "");
      isComplexFieldExists = true;
    }
  }
  if(isComplexFieldExists){
     db.ExcelPageConfiguration.updateOne({_id:doc._id},
```

```
            {$set:
                {"columnConfigurations":columnConfigurations}
        });
    }
});
```

ii. Remove the "**import com.avanseus.database.mongo.MongoPersistenceManager;\n**" statement from **generatedCode** field in Postprocessor collection and from **generatedJavaCode** field in Preprocessor collection.

Run the below MongoDB scripts to apply the above mentioned changes:

```
//For Postprocessor collection
db.Postprocessor.find({}).forEach(function(doc){
  var generatedCode = doc.generatedCode;
  generatedCode = generatedCode.replace(/import com.avanseus.database.mongo.MongoPersistenceManager;\n/g,
"");
  db.Postprocessor.updateOne({_id:doc._id},
     {$set:
          {"generatedCode":generatedCode}
     });
});

//For Preprocessor collection
db.Preprocessor.find({}).forEach(function(doc){
  var generatedJavaCode = doc.generatedJavaCode;
  generatedJavaCode = generatedJavaCode.replace(/import
com.avanseus.database.mongo.MongoPersistenceManager;\n/g, "");
  db.Preprocessor.updateOne({_id:doc._id},
     {$set:
          {"generatedJavaCode":generatedJavaCode}
     });
});
```

iii. Remove the "**import com.avanseus.database.mongo.MongoPersistenceManager;\n**" statement from **templateCodeSnippet** field in EventFileFormatTemplate, PostProcessorTemplate, PreProcessorTemplate and ExcelPageConfigurationTemplate collection.

Run the below MongoDB scripts to apply the above mentioned changes:

```
//For EventFileFormatTemplate collection
db.EventFileFormatTemplate.find({}).forEach(function(doc){
  var templateCodeSnippet = doc.templateCodeSnippet;
  templateCodeSnippet = templateCodeSnippet.replace(/import
com.avanseus.database.mongo.MongoPersistenceManager;\n/g, "");
  db.EventFileFormatTemplate.updateOne({_id:doc._id},
     {$set:
          {"templateCodeSnippet":templateCodeSnippet}
     });
});

//For PostProcessorTemplate collection
db.PostProcessorTemplate.find({}).forEach(function(doc){
  var templateCodeSnippet = doc.templateCodeSnippet;
  templateCodeSnippet = templateCodeSnippet.replace(/import
com.avanseus.database.mongo.MongoPersistenceManager;\n/g, "");
  db.PostProcessorTemplate.updateOne({_id:doc._id},{$set:{"templateCodeSnippet":templateCodeSnippet}});
});

//For PreProcessorTemplate collection
db.PreProcessorTemplate.find({}).forEach(function(doc){
  var templateCodeSnippet = doc.templateCodeSnippet;
```

```
  templateCodeSnippet = templateCodeSnippet.replace(/import
com.avanseus.database.mongo.MongoPersistenceManager;\n/g, "");
  db.PreProcessorTemplate.updateOne({_id:doc._id},{$set:{"templateCodeSnippet":templateCodeSnippet}});
});

//For ExcelPageConfigurationTemplate collection
db.ExcelPageConfigurationTemplate.find({}).forEach(function(doc){
  var templateCodeSnippet = doc.templateCodeSnippet;
  templateCodeSnippet = templateCodeSnippet.replace(/import
com.avanseus.database.mongo.MongoPersistenceManager;/g, "");

db.ExcelPageConfigurationTemplate.updateOne({_id:doc._id},{$set:{"templateCodeSnippet":templateCodeSnipp
et}});
});
```

2.6 Changes related to PredictedFault collection

New fields, "repetitiveAlarms" & "isRepetitive" are added to PredictedFault JSON for the VBI query "display repeat alarm prediction".

Every day, after prediction these two fields will get populated in the PredictedFault based on the ticket status. For this "updatePredictedFaultsWithRepetitiveAlarms", api (under "RepetitiveAlarmsAction.java") to be called from PredictionNodeMessageProcessor after completing the predictions.

2.7 Changes related to Employee Access collection.

New field 'isNewUser' added to Employee Access JSON and changed the type of existing expiryDate filed to Date from String and capitalised the value of status field.

For the existing entries run the below Mongo script to update the JSON.

```
function stringToDate(_date,_format,_delimiter){
        var formatLowerCase=  format.toLowerCase();
        var formatItems=formatLowerCase.split(_delimiter);
        var dateItems=_date.split(_delimiter);
        var monthIndex=formatItems.indexOf("mm");
        var dayIndex=formatItems.indexOf("dd");
        var yearIndex=formatItems.indexOf("yyyy");
        var month=parseInt(dateItems[monthIndex]);
        month-=1;
        var formatedDate =
              new Date(dateItems[yearIndex],month,dateItems[dayIndex]);
        return formatedDate;
}


db.EmployeeAccess.find({}).forEach(function(doc){
  var expiryDate = stringToDate(doc.expiryDate,"dd/MM/yyyy","/");
  var status = doc.status.toUpperCase();
  var isNewUser = false;
  db.EmployeeAccess.updateOne({_id:doc._id},
  {$set:
      {"expiryDate":expiryDate,
       "status":status,
       "isNewUser":isNewUser
      }
  });
});
```

2.8 Changes related to Filter Configuration.

i. PredictionKeyFilter JSON structure has been modified from the existing one. Sample JSON would look like below

```
{
   "_id" : ObjectId("5d5b9f1cf08c9557b8096a07"),
   "keyType" : "Cause",
   "keyValue" : "BOARD HARDWARE FAULT (SAMPLE)",
   "predictionRule_id" : "5746ab0a66bba21bf99dc004"
}
```

Run the below mongo script to apply the changes

```
db.PredictionKeyFilter.find({}).forEach(function(doc){
 if(doc.filterKeyComponents != undefined){
   var filterKeyComponents = doc.filterKeyComponents;
   var keyType = filterKeyComponents[0].type;
   var keyValue = filterKeyComponents[0].value;
   var predictionRule_id = doc.predictionFilter_id;
   db.PredictionKeyFilter.updateOne({_id:doc._id},
    {$set:
      {
       "keyType":keyType,
       "keyValue":keyValue,
       "predictionRule_id":predictionRule_id
      },
     $unset:
       {
        "filterKeyComponents":"",
        "predictionFilter_id":""
       }
   });
 }
});
```

After running above script, copy the object id from PredictionFilterRule collection and set its value for predictionRule_id in PredictionKeyFilter collection.

```
db.PredictionKeyFilter.updateMany({},{$set: {predictionRule_id:<PredictionFilterRule object id>}});
```

ii. New collection 'PredictionFilterRule' has to be created for the Filter Configuration usage.

Run the generateFilterKeys.jsp which is under jsp directory of CAN. This will create the PredictionFilterRule collection. Pass the parameter 'filterId' with value **5746ab0a66bba21bf99dc004** to this jsp.

Note:
Before running the above jsp, make sure that this API, **predictionKeyFilterManager.savePredictionKeyFilter(predictionKeyFilter)** is commented out in the JSP.

https://<domain>/CAN/jsp/generateFilterKeys.jsp?filterId= **5746ab0a66bba21bf99dc004**

After doing the above changes i and ii, either restart the tomcat or compile the rules in the Filter Configuration screen in order to apply the above changes in the application.

If there are any specific columns or entries in the databases to be retained or modified, delivery developers should take care of that.

Note:

Domain, networkType, causeCategory, serviceAffecting & priority fields are mandatory in Cause collection. Office Code priority field is mandatory in Office Code collection. Office code is mandatory entity with CAN 5.0 in Alarm, AlarmGUI and PredictedFault collection.

3. Update the database with JSON structure changes to the collections

Please take the backup of the existing collections as mentioned below:

1. If there are any specific configurations that are needed after the upgrade.
2. The data dump JSON files are available in the "MasterTables" directory of the release repository. WeatherStatusResponseCode, DomainIcon and PredictionFilter dump is required. Use the dump to populate these tables.
3. After taking the backup of the above collections, run the import script (shown below):

   Delete the existing WeatherStatusResponseCode, DomainIcon and PredictionFilter collection and import the updated collection from the MasterTables directory.

```
1. mongoimport --db <db> --collection WeatherStatusResponseCode --username <username> --password <password> --file WeatherStatusResponseCode.json
2. mongoimport --db <db> --collection DomainIcon --username <username> --password <password> --file DomainIcon.json
3.mongoimport --db <db> --collection PredictionFilter --username <username> --password <password> --file PredictionFilter.json
4.mongoimport --db <db> --collection PredictionFilterTemplate --username <username> --password <password> --file PredictionFilterTemplate.json
```

4. Update the database with new collections

Put the new collections in the database. The new collections are available in the "MasterTables" directory of the release repository. The list of collections to be imported into the database for the new features are given below:

- RemedyConfig
- RemedyFieldValue
- PredictedFaultToTicketMappingTemplate
- SplunkFields
- VBIAnnotations

Please run the import script (shown below) to load the new tables:

```
1. mongoimport --db <db> --collection RemedyConfig --username <username> --password <password> --file RemedyConfig.json
2. mongoimport --db <db> --collection RemedyFieldValue --username <username> --password <password> --file RemedyFieldValue.json
3. mongoimport --db <db> --collection PredictedFaultToTicketMappingTemplate --username <username> --password <password> --file PredictedFaultToTicketMappingTemplate.json
4. mongoimport --db <db> --collection SplunkFields --username <username> --password <password> --file SplunkFields.json
5. mongoimport --db <db> --collection VBIAnnotations --username <username> --password <password> --file VBIAnnotations.json
```

5. Addition of new config entries in Config collection

Add the below config entries in Config collection. These config entries are related to:

5.1 Cause


```
{
    "_id" : ObjectId("600e736ccc111a65ced822bf"),
    "key" : "domain-networkType",
    "value" : "[{\"domain\":\"CORE\",\"networkType\":[\"PS-CORE\",\"CS-
CORE\"]},{\"domain\":\"ACCESS\",\"networkType\":[\"2G\",\"3G\",\"4G\"]},{\"domain\":\"TRANSPORT\",\"networkTyp
e\":[\"CEN\",\"MPLS\"]}]",
    "title" : "Network Type",
    "modules" : "CAN",
    "type" : "String",
    "group" : "Cause",
    "displayLabel" : "config.domain-networkType",
    "fieldName" : "domain-networkType",
    "tableName" : "Cause",
    "modifiedOn" : ISODate("2021-01-29T07:15:40.733+0000")
}
{
    "_id" : ObjectId("5e8c4fbf32a89f20971a9714"),
    "key" : "priority",
    "title" : "Priority",
    "value" : "false",
    "type" : "boolean",
    "modules" : "CAN",
    "group" : "Cause",
    "displayOrder" : NumberInt(6),
    "tableName" : "Cause",
    "fieldName" : "priority",
    "displayLabel" : "config.priority",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.798+0000")
}
{
    "_id" : ObjectId("5e8c501032a89f20971a9715"),
    "key" : "serviceAffecting",
    "title" : "Affect Service",
    "value" : "false",
    "type" : "boolean",
    "modules" : "CAN",
    "group" : "Cause",
    "displayOrder" : NumberInt(6),
    "displayLabel" : "config.serviceAffecting",
    "tableName" : "Cause",
    "fieldName" : "serviceAffecting",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.792+0000")
}
{
  "_id" : ObjectId("5e8c507532a89f20971a9716"),
  "key" : "causeCategory",
  "value" : "INFRA,HARDWARE,TEST",
  "title" : "Category",
  "displayOrder" : NumberInt(6),
  "displayLabel" : "config.causeCategory",
  "group" : "Cause",
  "type" : "String",
  "noOfDefaults" : 2.0,
  "enumName" : "com.avanseus.model.can.CauseCategory",
  "modules" : "CAN",
  "modifiedOn" : ISODate("2020-11-19T13:51:36.836+0000"),
  "tableName" : "Cause",
  "fieldName" : "causeCategory",
  "valuePossibilities" : [
    {
      "valueDisplayLabel" : "config.Infra",
      "valueTitle" : "INFRA",
      "actualValue" : "INFRA"
    },
```

```
    {
      "valueDisplayLabel" : "config.Hardware",
      "valueTitle" : "HARDWARE",
      "actualValue" : "HARDWARE"
    }
  ]
}
```

5.2 VBI

```
{
  "_id" : ObjectId("5e40fef6cbaaab0e1c1d0177"),
  "key" : "noOfRecords.networkStatus",
  "value" : "50"
}
{
  "_id" : ObjectId("5e53d2f1cbaaab26d7e8d5f1"),
  "key" : "vbi"
  "modules" : "CAN",
  "valuePossibilities" : [
    {
      "roleCategories" : [
        "SUPER_ADMIN",
        "ADMIN",
        "CIRCLE_MANAGER",
        "ZONE_LEAD",
        "OTHERS"
      ]
    },
    {
      "minThresholdForRejection" : 0.75,
      "minThresholdForSuggestion" : 0.82,
      "maxThresholdForMinConfidence" : 0.9,
      "maxThresholdForAveConfidence" : 0.88,
      "positive_negative_difference_threshold" : 0.03,
      "numberOfMatchingReq" : NumberInt(3),
      "languageReq" : "en-US"
    }
  ]
}
```

5.3 CKR

```
{
  "_id" : ObjectId("5eb303a8a7986c115cb7a2c0"),
  "key" : "countryCode",
  "value" : "IND",
  "modules" : "CAN",
  "type" : "String",
  "title" : "Country Code",
  "group" : "Knowledge repository",
  "displayLabel" : "config.countryCode",
  "modifiedOn" : ISODate("2020-08-19T13:37:29.831+0000")
}
{
  "_id" : ObjectId("5eb303a8a7986c115cb7a2c9"),
  "key" : "KnowledgeBasedRuleDiscovery",
  "value" : "false",
  "modules" : "CAN",
  "type" : "boolean",
  "group" : "Knowledge repository",
  "title" : "Rule",
  "displayLabel" : "config.knowledgeBasedRule",
  "modifiedOn" : ISODate("2020-08-19T13:37:29.760+0000")
}
```

5.4 Performance Counter Predictions


```
{
    "_id" : ObjectId("5ee07dd6a93a4b1f75a40f88"),
    "key" : "healthIndexOffset",
    "value" : "1",
    "modules" : "CAN",
    "type" : "String",
    "title" : "Offset",
    "displayLabel" : "config.healthIndexOffset",
    "group" : "Health Index",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.791+0000")
}
{
    "_id" : ObjectId("5ee07de8a93a4b1f75a40f89"),
    "key" : "pmCounterPredictionInterval",
    "value" : "4",
    "modules" : "CAN",
    "title" : "PM Prediction interval",
    "max" : 14.0,
    "min" : 1.0,
    "type" : "Int",
    "group" : "Performance Counter",
    "displayOrder" : NumberInt(3),
    "displayLabel" : "config.pmCounterPredictionInterval",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.782+0000")
}
{
    "_id" : ObjectId("5ee07dfea93a4b1f75a40f8a"),
    "key" : "pmCounterBitSequenceLength",
    "title" : "PM Bit Sequence Length",
    "value" : "60",
    "modules" : "CAN",
    "max" : 500.0,
    "min" : 50.0,
    "type" : "Int",
    "group" : "Performance Counter",
    "displayOrder" : NumberInt(4),
    "displayLabel" : "config.pmCounterBitSequenceLength",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.824+0000")
}
{
    "_id" : ObjectId("5ee07db7a93a4b1f75a40f87"),
    "key" : "healthIndexScalingFactor",
    "value" : "1",
    "modules" : "CAN",
    "type" : "String",
    "title" : "Scaling factor",
    "displayLabel" : "config.healthIndexScalingFactor",
    "group" : "Health Index",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.823+0000")
}
{
    "_id" : ObjectId("5ee07e15a93a4b1f75a40f8b"),
    "key" : "pmCounterSlotLength",
    "value" : "8",
    "modules" : "CAN",
    "max" : 300.0,
    "min" : 1.0,
    "title" : "PM Slot Length",
    "type" : "Int",
    "group" : "Performance Counter",
    "displayOrder" : NumberInt(4),
    "displayLabel" : "config.pmCounterSlotLength",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.842+0000")
```

```
}
{
    "_id" : ObjectId("5f0f4dc4d238e455efb5f6fa"),
    "key" : "meanClosedDaysThreshold",
    "value" : "1"
}
{
    "_id" : ObjectId("5f1ff15f32a89f3ba5efb855"),
    "key" : "predictionType",
    "value" : "KPI",
    "title" : "Prediction Type",
    "displayOrder" : NumberInt(6),
    "group" : "Performance Counter",
    "type" : "DropDown",
    "enumName" : "com.avanseus.model.can.PredictionType",
    "modules" : "CAN",
    "displayLabel" : "config.predictionType",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.784+0000")
}
{
    "_id" : ObjectId("5f1fec7032a89f3ba5efb854"),
    "key" : "healthIndexWarningLevel",
    "value" : "60",
    "modules" : "CAN",
    "max" : 100.0,
    "min" : 1.0,
    "title" : "Warning Level",
    "type" : "Int",
    "group" : "Health Index",
    "displayOrder" : NumberInt(4),
    "displayLabel" : "config.healthIndexWarningLevel",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.830+0000")
}
{
    "_id" : ObjectId("5f1fec2032a89f3ba5efb853"),
    "key" : "healthIndexCriticalLevel",
    "value" : "20",
    "modules" : "CAN",
    "max" : 100.0,
    "min" : 1.0,
    "title" : "Critical Level",
    "type" : "Int",
    "group" : "Health Index",
    "displayOrder" : NumberInt(4),
    "displayLabel" : "config.healthIndexCriticalLevel",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.845+0000")
}
{
    "_id" : ObjectId("5f1fe96032a89f3ba5efb851"),
    "key" : "pmCounterSlotDurationInMins",
    "value" : "15",
    "modules" : "CAN",
    "type" : "String",
    "title" : "Data Availability(Mins)",
    "displayLabel" : "config.pmCounterSlotDurationInMins",
    "group" : "Performance Counter",
    "modifiedOn" : ISODate("2020-08-19T13:37:29.811+0000")
}
{
    "_id" : ObjectId("5d14d778d238fc9ad9db0e7d"),
    "key" : "KPI Level",
    "value" : "Equipment",
    "title" : "Performance KPI Level",
    "group" : "Performance KPI",
    "type" : "DropDown",
    "enumName" : "com.avanseus.model.can.KpiLevel",
    "modules" : "CAN",
```



```
  "modifiedOn" : ISODate("2020-02-25T11:38:18.795+0000"),
  "displayOrder" : 1.0,
  "displayLabel" : "settings.performanceKPILevel.defaultRepresentation",
  "valuePossibilities" : [
    {
      "valueDisplayLabel" : "settings.performanceKPILevel.defaultRepresentationOption1",
      "valueTitle" : "Equipment",
      "actualValue" : "Equipment"
    },
    {
      "valueDisplayLabel" : "settings.performanceKPILevel.defaultRepresentationOption2",
      "valueTitle" : "EquipmentComponent",
      "actualValue" : "Equipment Component"
    }
  ]
}
{
  "_id" : ObjectId("5f10845f0b79c307a8541b2f"),
  "key" : "pmCounterMatchSlots",
  "value" : "1"
}
{
  "_id" : ObjectId("5f325ba405405d28bc0c5c92"),
  "key" : "minimumOnesForSuperposition",
  "value" : "1"
}
```


5.5 Weather Report


```
{
  "_id" : ObjectId("5f324cab0fb48e1191468695"),
  "isWeatherReportRequired" : "false"
}
```


6. Storing Passwords in Encrypted format in Database

In CAN 5.0, all the passwords which are stored in the config.properties must be encrypted. Please refer to supporting document which comes along with this document for complete details.

7. Compare the existing config entries with the config entries under master tables repository

Compare the existing config entries with the config entries under master tables repository for the following keys.


```
1. "key" : "recommendations"
2. "key" : "predictionMatchSlots"
3. "key" : "bitThreshold"
4. "key" : "default_fault_representation"
5. "key" : "KPI Level"
6. "key" : "sparsityMultiplier"
7. "key" : "vbi"
8. "key" : "causeCategory"
```


Update all these entries with the latest config JSON which is under master tables but retain the config values if in case you are using any of these properties in your deployment.

8. Update the new build of CAN and CAS in the tomcat server

A. Delete the earlier build from tomcats “webapps” directory. (Both war and the directory)
B. Take a new build of CAN and CAS from 5.0 release and copy to “webapps” directory.

C. Run the tomcat servers.