CAN INSTALLATION AND CONFIGURATION

Version 5.0

AUGUST 10, 2020

AVANSEUS TECHNOLOGIES PVT. LTD.

Table of Contents

1. Initial Configuration Steps

1. Setup the Open-DS, VBI python application & Mongo-DB. If required, refer vendor documentation.

2. Setup the tomcats for CAN & CAS.

3. Configure the “config. properties” file inside tomcat home directories for each module based on the environment i.e. IP’s & ports. See the sample content below:

```
#Domain details
avanseus.app.cas.domain=10.2.2.100:8001
avanseus.app.can.domain=127.0.0.1:8081
avanseus.app.ckr.domain=127.0.0.1:8082

#VBI application info
avanseus.vbi.ip=192.168.1.104
avanseus.vbi.port=12001

#Domain protocol
avanseus.protocol=https

#Host details
avanseus.app.cas.host=10.2.2.100:8001
avanseus.app.can.host=10.8.0.6:8080
avanseus.app.ckr.host:127.0.0.1:8082

#MongoDb configuration
avanseus.mongodb.ip=10.2.2.100
avanseus.mongodb.port=27017
avanseus.mongodb.username=<username>
avanseus.mongodb.password=<password>
avanseus.mongodb.dbName=<dbname>

#Note: This configuration is needed to get admin rights.
avanseus.mongodb.admin.username=<admin_username>
avanseus.mongodb.admin.password=<admin_password>
avanseus.mongodb.admin.dbName=<admin_dbname>

#log configuration
avanseus.log.path=/Users/avanseustechnologies/tomcat8/logs1/
avanseus.log.thread.poolsize=20

#LDAP server configuration
avanseus.ldap.organisation=employee
avanseus.ldap.domain=avanseus
avanseus.ldap.url1=ldap://10.2.2.101:1389
avanseus.ldap.url2=ldap://10.2.2.101:1389
avanseus.ldap.userDn=cn=Directory Manager
avanseus.ldap.password=password

#NAS path
```

```
avanseus.app.nas.path=/Users/avanseustechnologies/nasmount/

#Email Config
avanseus.email.fromId=can.admin@avanseus.com
avanseus.email.pwd=<password>
avanseus.email.host=<email_server>
avanseus.email.fromName=Admin
avanseus.email.port=587

#Cluster node configuration
avanseus.predictionNode.name=nodeA
```

4. Copy the contents inside “nasPath” directory which is inside “MasterTables” under nasmount path in the deployment.

5. Create a user in MongoDB and import the collections that are available in “MasterTables” directory of release repository. The collections and their purposes are as follows:

Entity name	Purpose
5bed7c01fd9b9d519fedd798, 5bed7c22fd9b9d519fedd79b, 5d0ce5a1cbaaab22bc8aa286, 5bed7f6afd9b9d519ffb5776, Resource & ResourceEntity	Contains default data for resource file load screen
Cause	Default alarm causes
AlertEmailConfiguration	Contains initial data for Notification handler screen
DefaultComplexCodeROE	Contains default configuration for ROE screen
DomainIcon	Contains default icon paths for default domains
Config	Generic configuration table
EmployeeAccess	Has the initial user id as 'canadmin'.
EntityFilterMaster, EntityFilterMasterQuery & EquipmentIcon	Contains default values for Cross domain correlation screen
EquipmentComponent	Default equipment components
EventFileFormat	Contains default data for Parser screen
EventFileFormatTemplate	Has master code template used for parser
ExcelPageConfiguration	Contains default data for Excel page configuration screen
ExcelPageConfigurationTemplate	Has master code template used for excel report generation
ExcelReportConfiguration	Contains default data for Excel report configuration screen
FieldLearntCauses	Contains default data of Root cause learnt from the field
FileCollectionConfiguration	Contains default data for file collection screen

Entity name	Purpose
PostPredictionProcess	Contains default data for Post prediction process screen
PostProcessorTemplate	Has master code template used for parser's post-processor
Postprocessor	Contains default data for post processor screen
PredictionAssignmentPolicy	Contains default data for prediction assignment policy screen
PreProcessorTemplate	Has master code template used for parser's pre-processor
PredictionFilter	Has master code logic for prediction filtration
PredictionFilterTemplate	Has master code template for prediction filtration
PredictionKeyFilter	Contains default data for prediction filter screen
PredictionNode	Contains default data for Prediction nodes. Default is 3 nodes as of now.
PredictedFaultToTicketMappingTemplate	Contains template needed to compile PredictionToTicketMapping code
Preprocessor	Contains default data for pre-processor screen
RemedyConfig	Contains remedy config values such as cron , mean threshold closed days
RemedyFieldValue	Contains fieldId and field names of remedy internal fields
RoeConfigTemplate	Has master code template for RoE configuration
RoeWeightageConfig	Contains default data for RoE screen
Role	Has initial role for "canadmin". Only Super Admin role would be present in Role table by default.
RootCauseAnalysis & RootCauseAnalysisEntity	Contains default data for Root cause analysis configuration screen
SplunkFields	Has splunk search job related fields and their default values
TechnicalAnalysedCauses	Contains default data of Root cause learnt from technical analysis
VBIAnnotations	Contains necessary information for VBI module related to possible questions voice based module accepts
WeatherStatusResponseCode	Contains weather codes, weather description, color code and image icon path

6. Also update the nasmount path in few DB collections by executing the below script.

```
var nasPath = "/home/user"

db.getCollection('5d23282ebb19a8c46c88c105').update({},{$set:{"filePath":nasPath+"/CAN/Resources/RANSharedSites/Report_SSI_H3G.XLS"}}, {multi:true})
db.getCollection('5d23285dbb19a8c46c88c128').update({},{$set:{"filePath":nasPath+"/CAN/Resources/PlannedWorks/Planned 01.05-05.11.2018.xlsx"}}, {multi:true})
```

```
db.getCollection('5d232805bb19a8c46c88c0f1').update({},{$set:{"filePath":nasPath+"/CAN/Resources/LongPendingTickets/Sol.xlsx"}}, {multi:true})
db.getCollection('5d232845bb19a8c46c88c117').update({},{$set:{"filePath":nasPath+"/CAN/Resources/SitePriority/Localization_sites.xls"}}, {multi:true})
db.getCollection('FieldLearntCauses').find({}).forEach(function(e){db.getCollection('FieldLearntCauses').update({filePath:e.filePath},{$set:{filePath:nasPath+e.filePath}})})
db.getCollection('PostPredictionProcess').update({"classFilePath":"/CAN/Generated_Dir"},{$set:{"classFilePath":nasPath+"/CAN/Generated_Dir"}},{})
db.getCollection('PostPredictionProcess').update({"javaFilePath":"/CAN/Generated_Dir/postPredictionProcessUploadedFile"},{$set:{"javaFilePath":nasPath+"/CAN/Generated_Dir/postPredictionProcessUploadedFile"}},{})
db.getCollection('ResourceEntity').find({}).forEach(function(e){db.getCollection('ResourceEntity').update({filePath:e.filePath},{$set:{filePath:nasPath+e.filePath}})})
db.getCollection('RootCauseAnalysisEntity').find({}).forEach(function(e){db.getCollection('RootCauseAnalysisEntity').update({filePath:e.filePath},{$set:{filePath:nasPath+e.filePath}})})
db.getCollection('TechnicalAnalysedCauses').update({"filePath":"/CAN/RootCauseAnalysis/technicalAnalysis/SampleFileForTechnicalAnalysis.xlsx"},{$set:{"filePath":nasPath+"/CAN/RootCauseAnalysis/technicalAnalysis/SampleFileForTechnicalAnalysis.xlsx"}},{multi:true})
```

In the above script, replace the nasmount path value with respective nasmount directory of deployment environment.

7. Deploy the CAN, CAS & GSC

8. Configure the file collection & parser.

2. Manual Verification Steps (Using Utilities)

1. Run the file collection & parsing: http://<domain>/CAN/pickupData.jsp
   This step is to load the data.

2. Create the "PredictionKeyFilter" table entries for all causes having default rule (0, 0):
   http://<domain>/CAN/createFilterRule.jsp?filterId=5746ab0a66bba21bf99dc004.

3. Create the “PredictionNode” table based on the number of nodes that are needed for distribution.

_id	priority	name	host	port
58a27baefa2...	2	nodeA	10.2.2.80	6768
5acc826aa58...	1	nodeB	10.2.2.80	6769

4. Create the “PredictionAssignmentPolicy” table based on the cause for prediction distribution across tomcats:
   http://<domain>/CAN/generateClusterAssignmentPolicy.jsp?nodes=<no_of_nodes>&field=<reference_field_from_TT_table>
   Eg: *http://<domain>/CAN/generateClusterAssignmentPolicy.jsp?nodes=2&field=cause_id*

5. Running clustering:
   http://<domain>/CAN/jsp/cluster.jsp
   Eg: http://127.0.0.1:8081/CAN/jsp/cluster.jsp

6. Running prediction:
   http://<domain>/CAN/jsp/predict.jsp?time=<dd-MM-yyyy>,<dd-MM-yyyy>....
   Eg: http://127.0.0.1:8081/CAN/jsp/predict.jsp?time=08-11-2017,09-11-2017

7. Generate the fault trace:
   http://<domain>/CAN/jsp/generateAndUpdateFaultTrace.jsp?historyDays=<days_to_be_updated_in_TT>&startDate=<dd-MM-yyyy>&endDate=<dd-MM-yyyy>
   Eg:*http://<domain>/CAN/jsp/generateAndUpdateFaultTrace.jsp?historyDays=30&startDate=11-01-2018&endDate=15-01-2018*

   NOTE: The "endDate" parameter is optional. It may not be given, if only one day fault trace has to be updated.

8. Generate the fault history:
   http://<domain>/CAN/jsp/updateFaultHistory.jsp?startDate=<dd-MM-yyyy>&endDate=<dd-MM-yyyy>
   Eg:*http://<domain>/CAN/jsp/updateFaultHistory.jsp?startDate=11-01-2018&endDate=15-01-2018*

   NOTE: The "endDate" parameter is optional. It may not be given, if only one day fault history has to be updated.

9. Configure the Excel report format & mailing list.

10. Generate the particular day's report:
    http://<domain>/CAN/reportEmail.jsp?time=<dd-MM-yyyy>

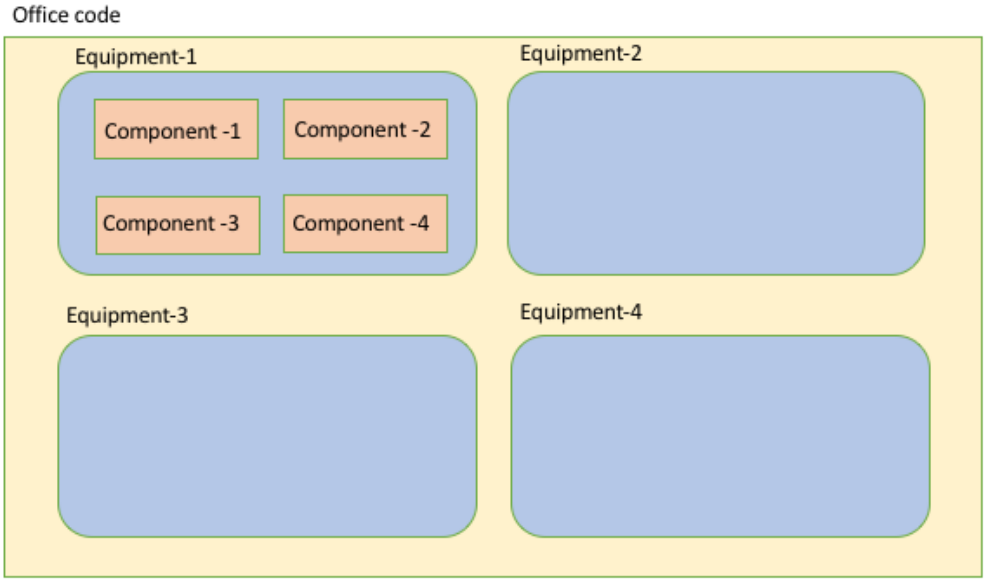
11. Generate the particular prediction window's matching report:
    http://<domain>/CAN/downloadMatchingReport.jsp?windowIdentifier=<time_in_milliseconds_of_the_first_day_of_prediction_window>

12. Formatted Predicted Fault table will be generated automatically after Prediction. If the table is not present in the database, then create that table manually by running:
    http://<domain>/CAN/jsp/formattedPredictedFaultTableCreation.jsp?startDate=<dd-MM-yyyy>&endDate=<dd-MM-yyyy>
    Eg:http://<domain>/CAN/jsp/formattedPredictedFaultTableCreation.jsp?startDate=20-05-2018&endDate=09-06-2018

    NOTE: To run the above URL, please make sure that all the code snippets written in the Page Configuration screen have been compiled and saved.

3. Notes on CAN Entity Hierarchy and Their Usage

A pictorial representation of Office code and internal components is shown below.



From the above picture, it is clear that a Site/Office code will have multiple equipment installed in it and each equipment will have multiple components. These components can be equipment's port, slot, rack etc.

In CAN application, we will have multiple entities that defines these Site components. The multiple entities are:

a. Office Code - DB collection would be Office Code
b. Equipment - DB collection would be Equipment
c. Equipment Component - DB collection would be Equipment Component

Above entities are made mandatory in the Parser screen for the Alarm configuration.

The table shown below contains different operations along with the Entity level.

Operation	Entity level
Prediction	Equipment component
Fault history	Equipment
Fault trace	Equipment
Clustering/Cross domain correlation	Office code
RC analysis	Equipment
ROE (Return on effort)	Equipment
Ticket history	Equipment