ERICSSON

ENM AGAT Setup and User Guide

User Guide

1/1553-6/AOM 901 151 Uen AN1

Contents

1 Introduction

This document describes the procedure to setup and execute the Automated Generic Acceptance Test (AGAT), CXP 901 6311, towards an ENM System.

The AGAT supports the following functions:

— Configuring test data

— Executing test suites containing test cases

— Storing and viewing the test results

Target Audience

This document is intended as an introduction to the test function for network operators, network and service planners, system engineers and administrators.

It is assumed that the user of this document has a basic knowledge of data communication and telecommunication.

2 AGAT Prerequisites

— Test Executor (TE) has been successfully deployed and is operational. To deploy and configure TE, refer to *Test Executor Configuration and Operation*, 1531-CNA 403 4011.

— There is at least 20 GB of free disk space.

— Java JRE and Java JDK are available.

If not, install version 1.8.0.121.

— There are ENM feature-specific prerequisites to be performed before running the AGAT tests.

- AMOS

  To launch an AMOS or Element Manager session towards Security Level 2 (SL2) for CPP nodes or Transport Layer Security (TLS) for nodes supporting ECIM, a user needs SSU credentials.

  To configure users, refer to the *ENM Network Security Configuration System Administrator Guide*, 2/1543-AOM 901 151-2.

- Release Independence

  To execute the Release Independence (RI) test suite, refer to the prerequisites of the Release Independence GAT.

- Software Hardware Manager

  To execute the Software Hardware Manager (SHM) test suites, a license file and upgrade software package is required for the node.

- FM

  To execute FM test suites, it is required to disable FMX rules and also to disable Auto-ack route in the system to obtain proper results.

  For MGW nodes, a PM subscription is mandatory to raise a test alarm.

— Tests with `write` access to a common file must not be executed simultaneously. The reason for this is to avoid unexpected on the fly file changes (while test is running). This rule applies even if the tests pertain to different test areas.

Example:

In the `Agat_AMOS.xml` suite, the `autoDisconnectInactiveSshSessionOnAmosAndScriptingVmsNew` test

case wants to access `/home/shared/common/sshd_config.properties` to get, modify, and put back the `sshd` timeout value. If this is done with more than one access it may return in assertion failure.

— FM RNL for Router 6672

Below pre-requisites should be configured on the node to get Unique License alarm :

- Node should have configured with L2VPN or L3VPN (Only one VPN configuration should be present).
- LKF should be installed on the Node. (LKF should include L2VPN or L3VPN key, according to which type of VPN will be configured.
- License should not be expired, Granted Capacity level should be more than 0.

3 Deploy Latest Testware and Dependencies

This section describes the steps involved to extract and deploy the AGAT testware successfully.

The steps in the following task proceed as follows:

— Extract the testware dependencies.

— Copy and run the setup script.

— Verify the contents of the local repository on the container.

Steps

Extract the Testware Dependencies.

1. Create the following folder structure:

```
[root@localhost~]# mkdir /root/agat/
```

2. Retrieve the ISO file containing the testware from the Software Gateway and store it in the `/root/agat/` folder.

3. Navigate to `/root/agat/` folder.

```
[root@localhost~]# cd /root/agat/
```

4. Mount the testware ISO with the following commands. This also unmounts any previously mounted ISO on `/root/.m2/repository`.

```
[root@localhost agat]# (mount | egrep '/root/.m2/repository') && umount /roo →
t/.m2/repository
[root@localhost agat]# mount -o ro,loop ERICenmagat_CXP9036311-x.x.x.iso /ro →
ot/.m2/repository
```

5. Confirm that the repository directory is not empty and that files were extracted to the location.

```
[root@localhost~]# ls /root/.m2/repository/
0_TestDataTemplateFiles           backport-util-concurrent  commons-chain     →
       commons-httpclient  de         javax      log4j     oro       xerces
agat_schedule_20180621181941.xml  c3p0                      commons-cli       →
       commons-io          dom4j      jaxen      logkit    rr_moved  xml-api →
s
antlr                             cglib                     commons-codec     →
       commons-lang        eu         jgraph     mysql     ru
aopalliance                       classworlds               commons-collecti →
ons    commons-logging     iaik       jline      nekohtml  sshtools
asm                               com                       commons-configur →
ation  commons-pool        io         joda-time  net       sslext
```

```
avalon-framework                commons-beanutils         commons-digester →
       commons-validator   javassist  junit      org       xalan
```

Copy and Run the Setup Script.

Note: If you are already using TDM, log on and export the `NodeData.sample` and `UsersToCreate.sample` data sources before running `setupTDM.sh`.

6. Copy the `setupTDM.sh` script from `/root/.m2/repository/0_TestDataTemplateFiles/` to `/root/agat/`.

```
[root@localhost agat]# cp /root/.m2/repository/0_TestDataTemplateFiles/setup →
TDM.sh /root/agat/
```

7. Give executable permission to the `setupTDM.sh` script.

```
[root@localhost agat]# chmod 777 setupTDM.sh
```

8. Run `setupTDM.sh` from `/root/agat/`. This script restarts the Docker container, creates the AGAT folder structure under `/root/agat`, copies all necessary files from the mounted ISO, and restores the TDM database to contain all the data sources.

```
[root@localhost agat]# ./setupTDM.sh
```

Verify the Contents of the Local Repository on the Container.

9. Navigate into the container with the following command:

```
[root@localhost~]# docker exec -it `docker ps -q -f 'NAME=taf_te_slave'` /bi →
n/bash
                    root@4c95c0c1d6e9:~#
```

10. Navigate to the `/root/.m2/repository/` directory on the container.

```
root@4c95c0c1d6e9:~# cd /root/.m2/repository/
```

11. Verify that the file structure and versions in this location on the container are the same on the localhost directory `/root/.m2/repository/`.

```
                    root@4c95c0c1d6e9:~# ls /root/.m2/repository/
                    0_TestDataTemplateFiles           backport-util-concurre →
nt  commons-chain           commons-httpclient  de          javax       log4j   →
   oro       xerces
                    agat_schedule_20180621181941.xml  c3p0                    →
    commons-cli             commons-io          dom4j       jaxen       logkit  →
   rr_moved  xml-apis
                    antlr                               cglib                  →
    commons-codec           commons-lang        eu          jgraph      mysql   →
   ru
                    aopalliance                         classworlds            →
    commons-collections     commons-logging     iaik        jline       nekohtm →
l  sshtools
                    asm                                 com                    →
    commons-configuration   commons-pool        io          joda-time   net     →
   sslext
```

```
                    avalon-framework                commons-beanutils       →
    commons-digester       commons-validator   javassist  junit      org      →
   xalan
```

12. Once confirmed, exit from the container.

```
root@4c95c0c1d6e9:~# exit
```

13. Copy the `suitesExtraction.sh` script from `/root/.m2/repository/0_TestDataTemplateFiles/` to `/root/agat/`.

```
cp /root/.m2/repository/0_TestDataTemplateFiles/suitesExtraction.sh /root/ag →
at
```

14. Give executable permission to the `suitesExtraction.sh` script.

```
chmod 777 suitesExtraction.sh
```

15. Run the `suitesExtraction.sh` script.

```
./suitesExtraction.sh
```

Option 1

— copy the file `agat_schedule_<timestamp>.xml` with the name `agat_schedule.xml` and execute script

```
./suitesExtraction.sh
```

This script creates the folder `/root/testdata/suites`, to which all suites XML files are copied.

Option 2

— Execute script with `agat_schedule_<timestamp>.xml` as parameter:

```
./suitesExtraction.sh agat_schedule_<timestamp>.xml
```

This script creates the folder `/root/testdata/suites`, to which all suites XML files are copied.

Note: The two options are equivalent: without parameters the name of the default file (agat_schedule.xml) is taken, otherwise the xml file, passed onto as parameter, is used.

16. Log on to the TDM UI application (`https://<TE address>:9443/`) with the user credentials added while configuring your LDAP user during the installation of TE.

17. Verify that all the data source files are properly restored.

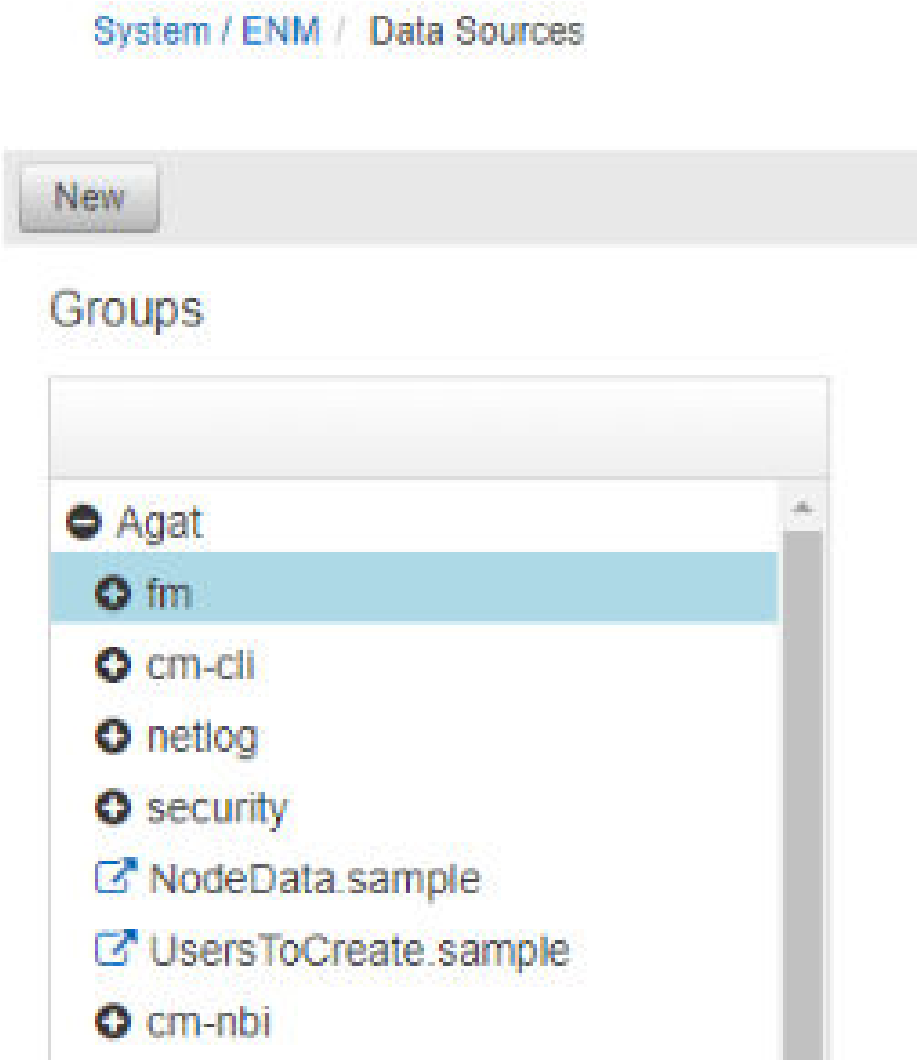


Figure 1 Restored Data Source Files

4 Test Configuration Files

> In order for the testware to run successfully, certain files need to be edited with environment-specific information.

This section describes each test configuration file and shows how to set up and configure the files.

Instructions of required changes are shown for the following files which need configuration to run the testware:

File	Description	Default File Name
Schedule XML	XML file containing information about the testware to be executed	`agat_schedule_yyyymmddhhmmss.xml`
Host properties	JSON file containing all necessary information about the ENM hosts belonging to the system under test	`host.properties.json`
Test properties	Properties file containing configuration data for the testware	`tdm.properties`

Data Source	Description	Data Source Name
Node data	TDM data source containing node test data that the testware uses for execution. Also known as a node data source.	NodeData.sample
User data	TDM data source containing user test data that the testware uses for execution. Also known as a user data source	UsersToCreate.sample

4.1 Schedule File

> The schedule file is an XML file containing information about the testware to be executed and the execution order.

By default, the schedule file is configured to run all tests in a particular order.

Note: The schedule file does not need to be edited. Exceptionally, it can be used to remove certain test suites according to special test requirements.

The schedule file name is `agat_schedule_yyyymmddhhmmss.xml`, where `yyyymmddhhmmss` is a date and time stamp.

The schedule file is located at `/root/agat/`. Check if the file is present with the command:

```
[root@localhost~]# ls /root/agat/
```

```
[root@atvts3081 agat]# ls /root/agat/
agat_schedule_20181214150138.xml  ERICenmagat_CXP9036311-1.68.1.3.iso
```

Figure 2 Example Schedule File Name and Location

```
<?xml version="1.0"?>
<schedule xmlns="http://taf.lmera.ericsson.se/schema/te"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://taf.lmera.ericsson.se/schema/te http://taf.lmera.ericsson.se/schema/te/schedule/xml">
    <item-group parallel="false">
        <item timeout-in-seconds="1800" stop-on-fail="false">
            <name>AGAT_PreCond</name>
            <component>com.ericsson.oss.rv:ERICTAFenmrv-scenarios-testware_CXP9031772:1.4.85</component>
            <suites>Agat_PreCond.xml</suites>
            <groups>ENM_EXTERNAL_TESTWARE</groups>
        </item>
        <item-group parallel="true">
            <item-group parallel="true">
                <item timeout-in-seconds="1800" stop-on-fail="false">
                    <name>CM_GAT</name>
                    <component>com.ericsson.oss.testware.cm:ERICTAFcmgattestware_CXP9034593:1.13.8</component>
                    <suites>Agat_CM.xml</suites>
                    <groups>ENM_EXTERNAL_TESTWARE</groups>
                </item>
                <item timeout-in-seconds="1800" stop-on-fail="false">
                    <name>Agat_Netlog</name>
                    <component>com.ericsson.oss.testware.netlog:ERICTAFnetlogtafsuitetestware_CXP9033258:1.21.3</component>
                    <suites>Agat_Netlog.xml</suites>
                    <groups>ENM_EXTERNAL_TESTWARE</groups>
                </item>
            </item-group>
            <item-group parallel="false">
                <item timeout-in-seconds="1800" stop-on-fail="false">
                    <name>Agat_AMOS</name>
                    <component>com.ericsson.oss.services.amos:ERICTAFamos_CXP9033072:1.28.4</component>
                    <suites>Agat_AMOS.xml</suites>
                    <groups>ENM_EXTERNAL_TESTWARE</groups>
                </item>
                <item-group parallel="false">
                    <item timeout-in-seconds="1800" stop-on-fail="false">
                        <name>Agat_FM</name>
                        <component>com.ericsson.oss.testware.fm:ERICTAFfmacceptancetestware_CXP9032937:1.41.2</component>
                        <suites>Agat_FM.xml</suites>
                        <groups>ENM_EXTERNAL_TESTWARE</groups>
                    </item>
                    <item timeout-in-seconds="1800" stop-on-fail="false">
                        <name>Agat_FM_Service</name>
                        <component>com.ericsson.oss.testware.fm:ERICTAFfmacceptancetestware_CXP9032937:1.41.2</component>
                        <suites>Agat_FM_Service.xml</suites>
                        <groups>ENM_EXTERNAL_TESTWARE</groups>
                    </item>
                </item-group>
            </item-group>
        </item-group>
    </item-group>
</schedule>
```

Figure 3 Example of Schedule File with a Few Suites Configured

The schedule XML file contains different suites for different applications and the core parameters to highlight in the schedule file are:

Parameter	Description
`parallel`	Determines if a suite can run in parallel during execution. In some cases, certain suites can run in parallel without having an impact on results, for example, PM.
`timeout-in-seconds`	The required time-out for a particular application suite to respond to a particular request. If there is no response from the application during this time period, the test suite fails and moves on to next suite.
`name`	The testware name.
`component`	The testware repository, product number, and version.
`suites`	The application suite name where each suite contains the test cases for each application. The test cases cannot be added or removed from suite name in runtime.

4.2 Host Properties File

> The host properties file contains all necessary information about the ENM hosts belonging to the system under test.

The host properties file is named `host.properties.json`. A template of `host.properties.json` file is located at `/root/agat/host.properties.json.sample`, as shown in the following figure:

```
[root@atvts3081 agat]# ls /root/agat/host.properties.json.sample
/root/agat/host.properties.json.sample
```

Figure 4 Template Location

Rename the host properties file from `/root/agat/host.properties.json.sample` to `/root/agat/host.properties.json`:

```
[root@localhost~]# mv /root/agat/host.properties.json.sample /root/agat/host.pro →
perties.json
```

Example of `host.properties.json` template file:

```
[
  {
    "ip": "<IP Address or hostname>",
    "hostname": "ms1",
    "type": "ms",
    "ports": {
      "ssh": 22
    },
    "users": [
      {
        "password": "<password>",
        "type": "admin",
        "username": "<username>"
      }
    ]
  },
  {
    "ip": "<IP Address or hostname>",
    "hostname": "haproxy_0",
    "type": "httpd",
    "ports": {
      "http": 80,
      "https": 443,
      "ssh": 22
    },
    "group": "haproxy"
  },
  {
    "ip": "<IP Address or hostname>",
    "hostname": "scripting_1",
    "type": "jboss",
    "ports": {
      "ssh": 22
    },
    "group": "scripting"
  },
  {
    "ip": "<IP Address or hostname>",
    "hostname": "scripting_2",
    "type": "jboss",
    "ports": {
```

```
      "ssh": 22
    },
    "group": "scripting"
  },
  {
    "ip": "<IP Address or hostname>",
    "hostname": "amos_1",
    "type": "jboss",
    "ports": {
      "ssh": 22
    },
    "group": "amos"
  },
  {
    "ip": "<IP Address or hostname>",
    "hostname": "amos_2",
    "type": "jboss",
    "ports": {
      "ssh": 22
    },
    "group": "amos"
  }
]
```

The host properties file contains environment-specific data, populate it before running any test and is composed by a list of objects. Each object contains the configuration of a VM.

Use all the public IP addresses of the VMs specified in the template when populating the "`ip`" property in an object.

4.3 Test Properties File

> The test properties file is a Java `.properties` file that is used to overwrite properties that are specified in the testware. This can be achieved by setting the same key with the new value.

The test properties file is named `tdm.properties`. A template of the `tdm.properties` file is located at `/root/agat/tdm.properties.sample`:

```
[root@atvts3179 ~]# ls /root/agat/tdm.properties.sample
/root/agat/tdm.properties.sample
```

Figure 5 tdm.properties Location

Copy the `/root/agat/tdm.properties.sample` file to `/root/agat/tdm.properties`:

```
[root@localhost~]# cp /root/agat/tdm.properties.sample /root/agat/tdm.properties
```

Figure 6 Example Snippet of the tdm.properties File

The necessary data to configure in the `tdm.properties` file are as follows:

Attribute	Description
`ci-user`	ENM test user with Administrator and SECURITY_ADMIN role permission. This user is responsible for the creation and deletion of other users used in the testware.
`ci-user-password`	ENM user (ci-user) password.
`filter.node`	Contains the node types that are used during the test, separated by semicolons. It filters the nodes, by node type, available in the `NodeData` datasource. For example, if ERBS and Radio nodes are used: filter.node=RadioNode;ERBS If the parameter is left empty, the testware only executes non-node-dependent test cases.
Node data properties for CM testware	Each node type has different parameters / settings that are required to run CM testware successfully. Template files only use ERBS and RadioNode as examples, so additional setup is required for other node types. See Appendix A - Add / Configure Datasources for CM on page 38 for more details when adding additional nodes.
`tdm.api.host`	TDM host where all the data sources required for the test case execution are present. Example: `https://<tdm.address>:9443/api`
`tdm.address`	The address of the host where the TE Docker containers are running.
`host.UiTestGrid.ip`	The address of the host where the TE Docker containers are running.

In order to execute the tests with a previous version of the `NodeData.Sample` file, use the following property:

```
dataprovider.nodesToAdd.version=<version>
```

Similarly, to use a previous version of `UsersToCreate.sample`, add the following property:

```
dataprovider.usersToCreateTemp.version=<version>
```

4.4 Node Data Configuration

> `NodeData.sample` is a TDM data source file that contains node test data used by the testware for execution.

`NodeData.sample` data source is located under **Agat**:

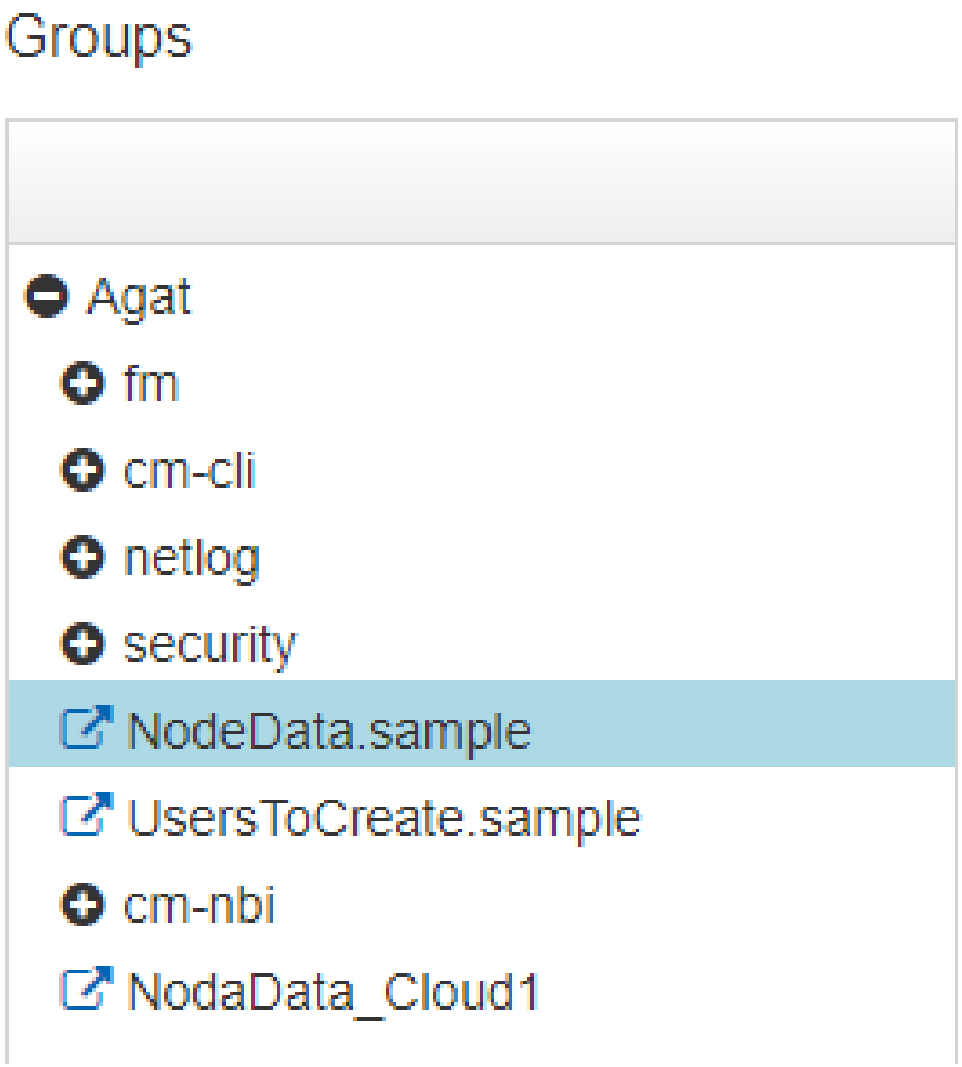


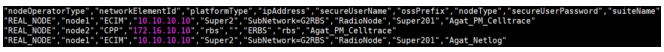
Figure 7 Agat → NodeData.Sample

If there is migration from remoteInfo to TDM, refer to Appendix E.

If you are already using TDM, refer to steps 2 and 3 in Appendix E.

For information on how to create and edit data sources in TDM, click the **Help** icon on the **TAF Test Data Management** page.

The table below is a description of all the parameters and required values for the `NodeData.sample` data source.

Parameter	Description
`ipAddress`	Represents the node IP address. The value must be known when configuring this column. It can be fetched with: `cmedit get <nodeName> CppConnectivityInformation.ipAddress`
`networkElementId`	Represents the Network Element ID of the node. The node name must be known when configuring this column. It can be fetched with the command: `cmedit get NetworkElement=<node name>`
`nodeOperatorType`	Represents the operator type, and must have the value REAL_NODE, as it is testware-dependent, otherwise it fails.
`nodeType`	Represents the node type. This value is used in the NodeType tab in the Allure report. It can be fetched with command: `cmedit get NetworkElement=<node name>` If there are ERBS and Micro_ERBS, the user enters those values in the NodeData data source.
`ossPrefix`	Represents the oss prefix of the node. The node name must be known when configuring this column. It can be fetched with command: `cmedit get NetworkElement=<node name>`
`platformType`	Represents the platform type of the node. The node name must be known when configuring this column. It can be fetched with command: `cmedit get NetworkElement=<node name>`
`secureUserName`	The secure user name of the node. The value must be known when configuring this column. It can be fetched with: `secadm credentials get --plaintext show --nodelist <nodeName>`
`secureUserPassword`	Represents the secure user password of the node. The value must be known when configuring this column. It can be fetched with: `secadm credentials get --plaintext show --nodelist <nodeName>`
`suiteName`	Column that associates nodes with a test suite. Runs specific test suites against selected node types. This is done by inserting the name of the suite, preconfigured on the scheduler.xml file, as the column value. For example, to run the Agat_PM_Celltrace suite against ERBS (node2) and Radio (node1) nodes, and Agat_Netlog suite against Radio (node1) node only, the entries in the CSV must be configured as shown in the figure below, and can be imported to the respective TDM data source.  Figure 8 Associating Nodes with a Test Suite

A set of optional parameters, required only for the `Agat_CM_DeleteNode` suite, are needed to recreate the node and depend on the node type. The set of parameters used when creating a node from the GUI must be used.

Table 1 Optional Parameters (Agat_CM_DeleteNode)

Parameter	Description
`normalUserName`	Normal user name for the node
`normalUserPasssword`	Normal password for the node
`rootUserName`	root user name for the node

Parameter	Description
rootUserPassword	root password for the node
port	Node port
transportType	Node transport type
snmpAgentPort	Node SNMP port
snmpVersion	Node SNMP version

By default, the suiteName column values are already included in the NodeData template data source which uses two fictional nodes (ERBS, RadioNode) as an example.

The suiteName column associates a node with a test suite. To run a test suite against a node, a line needs to be added with the test suite name in the suiteName column.

The table below highlights the suite names that can be used in the NodeData suiteName column. It also shows suites that are not node dependent, therefore not needed to be present on NodeData.

XML	suiteName	Configured	Node Dependent	Comment
Agat_UserSecurity.xml	Identity Management - Testware	N/A	No	This test suite does not use this feature, hence no entries are required in the NodeData data source.
Agat_SHM_Upgrade_Node_SW.xml	Agat_SHM_Upgrade_Node_SW	Yes	Yes	
Agat_SHM_Restorebackup.xml	Agat_SHM_Restoreback up	Yes	Yes	
Agat_SHM_Import_Install_LKF.xml	Agat_SHM_Import_Install_LKF	Yes	Yes	
Agat_SHM.xml	Agat_SHM	Yes	Yes	
Agat_Security_PKI.xml	Agat_Security_PKI	Yes	Yes	
Agat_RI_No_Remove_Support_for_a_Node_Version.xml	Agat_RI_No_Remove_Support_for_a_Node_Version	Yes	Yes	
Agat_RI.xml	Agat_RI	Yes	Yes	
Agat_PreCond.xml	AGAT_PreCond	Yes	Yes	
Agat_PM_STATISTICAL_SystemDefined.xml	Agat_PM_STATISTICAL_SystemDefined	Yes	Yes	
Agat_PM_CELLTRACE_CCTR.xml	Agat_PM_CELLTRACE_CCTR	Yes	Yes	
Agat_PM_STATISTICAL.xml	Agat_PM_STATISTICAL	Yes	Yes	
Agat_PM_CELLTRACE.xml	Agat_PM_CELLTRACE	Yes	Yes	
Agat_NHM.xml	Agat_NHM	Yes	Yes	
Agat_Netlog.xml	Agat_Netlog	Yes	Yes	
Agat_FM.xml	Agat_FM	Yes	Yes	
Agat_CM.xml	CM_GAT	N/A	N/A	This test suite does not use this feature, hence

XML	suiteName	Configured	Node Dependent	Comment
				no entries are required in the `NodeData` data source.
`Agat_CM_REST_NBI.xml`	Agat_CM_REST_NBI	N/A	N/A	This test suite does not use this feature, hence no entries are required in the `NodeData` data source.
`Agat_AMOS.xml`	Agat_AMOS	Yes	Yes	
`Agat_SHM_RestoreCV.xml`	Agat_SHM_RestoreCV	Yes	Yes	
`Agat_PM_CTUM.xml`	Agat_PM_CTUM	Yes	Yes	
`Agat_PM_GPEH.xml`	Agat_PM_GPEH	Yes	Yes	
`Agat_AMOS_UI.xml`	Agat_AMOS_UI	Yes	Yes	
`Agat_ELEMENT_MANAGER.xml`	Agat_ELEMENT_MANAGER	Yes	Yes	
`Agat_PM_STATISTICAL_UI.xml`	Agat_PM_STATISTICAL_UI	Yes	Yes	
`Agat_FM_UI.xml`	Agat_FM_UI	Yes	Yes	
`Agat_PM_CELLTRACE_CBS.xml`	Agat_PM_CELLTRACE_CBS	Yes	Yes	
`Agat_CM_DeleteNode.xml`	Agat_CM_DeleteNode	Yes	Yes	This suite is intrusive and it requires a test node configured in the NodeData data source.

Additional Information for Agat_PM_CELLTRACE.xml, Agat_PM_STATISTICAL.xml, and Agat_PM_GPEH.xml Suites

These three PM suites have a different behavior depending on the node setup in the NodeData, for example in the event where the setup contains a node type with only one node.

Example:

For one ERBS node, the expected test case behavior for that node type is that it will not run two test cases and the logs will show two failures as the suites are tested on only one node:

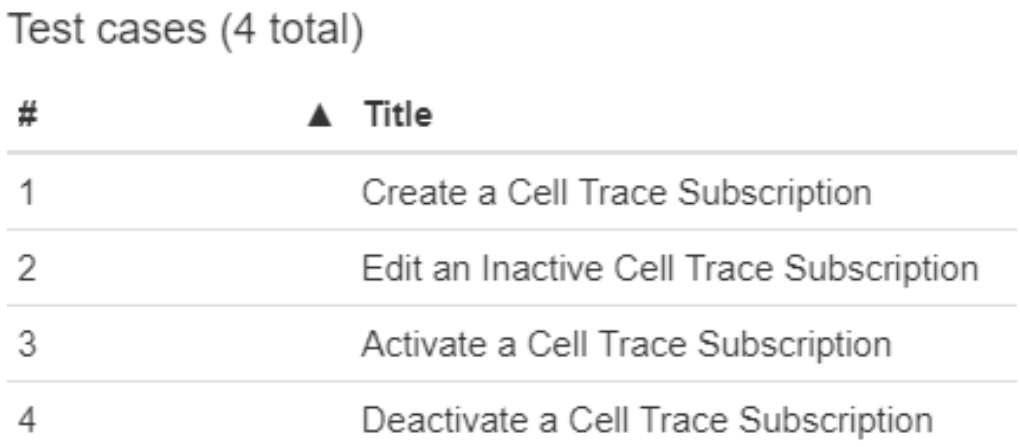


Figure 9 Example of Allure Report with Six Test Cases

```
Results :

Tests run: 6, Failures: 0, Errors: 0, Skipped: 0
```

Figure 10 Example of Allure Log

4.5 User Data Configuration

> The `UsersToCreate.sample` is a data source containing user test data that the testware uses for test execution.

A template of the `UsersToCreate.sample` data source is located in the **Agat** menu.

The table contains the description of all the parameters and required values for the `UsersToCreate.sample` data source.

Parameter	Data Type	Description
`firstName`	String	Represents the first name of the user. Can be any name according to the ENM policy.
`lastName`	String	Represents the last name of the user. Can be any name according to the ENM policy.
`password`	String	Represents the user password. Can be any password according to the ENM policy.
`roles`	String with roles inside ""	User roles that are needed to execute the testware, inside a "" and separated by commas. The `UsersToCreate` template data source comes with an example line that must be copied to each new user. This value cannot be edited and must be copied from the template. Otherwise it can cause testware malfunction.

Parameter	Data Type	Description
`email`	String	Represents the user email. Can be any email according to the ENM policy.
`enabled`	String `Enabled`	Flag to enable user. This value is always `Enabled`, otherwise it can cause testware malfunction.
`username`	String	Represents the username. Can be any name according to the ENM policy.

For information about how to create or edit a data source in TDM, click the **Help** icon on the **TAF Test Data Management** screen.

5 Additional Configuration for Specific Test Suites

This section details the specific configuration required for some test suites. If you are not planning to run any of the following test suites, ignore this section. Configuration is listed for the following test suites:

5.1 Perform Additional Configuration for Agat_SHM_Upgrade_Node_SW Suite

AGAT_SHM_Upgrade_Node_SW suite uses the `/root/testdata/shm/software` folder to search for software upgrade packages. The directory structure is created by `setupTDM.sh` script.

Steps

1. Verify that the folders have been created:

```
[root@atvts2267 ~]# ls /root/testdata/shm/software/
ERBS  MGW  RadioNode  RBS  RNC  SGSN
```

Example

To run the test suite against the RadioNode and ERBS nodes, upload the desired software packages in .zip format for each node type to the RadioNode and ERBS folders. Ensure that there is only one software upgrade file in each nodeType folder.

Also, verify that the SHM configuration is correct in the `/root/agat/tdm.properties` file, and add or edit the configuration if anything is missing or incorrect:

```
# SHM Specific Data
shm.software.path=/root/testdata/shm/software/
```

5.2 Perform Additional Configuration for AGAT_SHM_Import_Install_LKF Suite

AGAT_SHM_Import_Install_LKF suite uses the folder `/root/testdata/shm/license` to search for software upgrade packages. The directory structure is created by `setupTDM.sh` script.

Steps

1. Verify that the folders have been created:

```
[root@atvts2267 ~]# ls /root/testdata/shm/license/
ERBS  MGW  RadioNode_ RBS  RNC  SGSN
```

Example

To run the test suite against the RadioNode and ERBS nodes, upload the desired license key packages in .zip format for each node type to RadioNode and ERBS folders. Ensure that there is only one license key package file in each nodeType folder.

Also, verify that the SHM configuration is correct in the `/root/agat/tdm.properties` file, and add or edit properties if they are missing or incorrect:

```
# SHM Specific Data
shm.license.path=/root/testdata/shm/license/
```

5.3 Perform Additional Configuration for Agat_Netlog Suite

The Netlog dependent data sources are located under **netlog** in the Agat context menu, created from the mounted ISO by `setupTDM.sh` script.

Steps

1. Confirm that all the Netlog dependent data sources are present under **netlog**, for example:

Figure 11 Netlog Data Sources

2. Check if the following Netlog configuration properties are correct in the `/root/agat/tdm.properties` file:

```
#################################################################
#Netlog Specific Data
#################################################################
dataprovider.featureCommands.type=tdm
dataprovider.featureCommands.context=System/ENM
dataprovider.featureCommands.name=NetLog-CmdLog
```

The table below is a summary of the Netlog-dependent data sources with a short description of their usage:

TDM Data Source	Description
NetLog-CmdLog	This files contains a list of all node types supported by the testware and their log types.

5.4 Perform Additional Configuration for Agat_CM Suite

The CM-dependent TDM data sources are located under CM-CLI in the Agat context menu, created from the mounted ISO by the setupTDM.sh script:

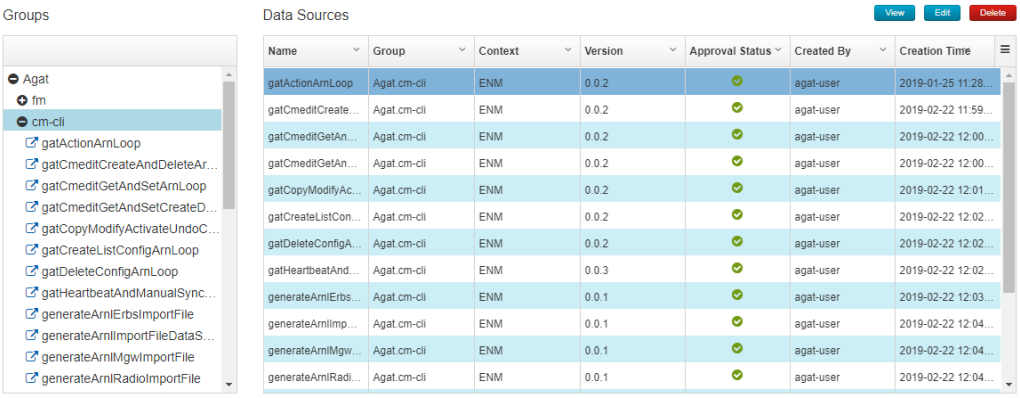


Figure 12 Example of TDM Data Source Names and Location

Prerequisites

Some of the tests must be run on a node. In this case, add the node information to the properties file. This node must also be in a synchronized state on the deployment.

Steps

1. Check if all the following data sources are present:
   - gatActionArnLoop
   - gatCmeditCreateAndDeleteArnLoop

- gatCmeditGetAndSetArnLoop
- gatCmeditGetAndSetCreateDeleteArnLoop
- gatConfigArnLoop
- gatCopyModifyActivateUndoConfigArnLoop
- gatCreateListConfigArnLoop
- gatDeleteConfigArnLoop
- gatHeartbeatAndManualSyncArnLoop
- gatImportArnLoop
- generateArnlImportFileDataSource
- generateArnl<NodeType>ImportFile (one for each node type configured, see Appendix A)
- verifyNonPersistedSetAndGetArnLoop

2. Check if the following CM configuration properties are correct in the `/root/agat/tdm.properties` file:

```
#############################################################
#CM-CLI Specific Data
#############################################################
dataprovider.gatHeartbeatAndManualSyncArnLoop.type=tdm
dataprovider.gatHeartbeatAndManualSyncArnLoop.context=System/ENM
dataprovider.gatHeartbeatAndManualSyncArnLoop.name=gatHeartbeatAndManualSync →
ArnLoop

dataprovider.gatCmeditGetAndSetArnLoop.type=tdm
dataprovider.gatCmeditGetAndSetArnLoop.context=System/ENM
dataprovider.gatCmeditGetAndSetArnLoop.name=gatCmeditGetAndSetArnLoop

dataprovider.gatCmeditCreateAndDeleteArnLoop.type=tdm
dataprovider.gatCmeditCreateAndDeleteArnLoop.context=System/ENM
dataprovider.gatCmeditCreateAndDeleteArnLoop.name=gatCmeditCreateAndDeleteAr →
nLoop

dataprovider.gatCreateListConfigArnLoop.type=tdm
dataprovider.gatCreateListConfigArnLoop.context=System/ENM
dataprovider.gatCreateListConfigArnLoop.name=gatCreateListConfigArnLoop

dataprovider.gatCopyModifyActivateUndoConfigArnLoop.type=tdm
dataprovider.gatCopyModifyActivateUndoConfigArnLoop.context=System/ENM
dataprovider.gatCopyModifyActivateUndoConfigArnLoop.name=gatCopyModifyActiva →
teUndoConfigArnLoop

dataprovider.gatDeleteConfigArnLoop.type=tdm
dataprovider.gatDeleteConfigArnLoop.context=System/ENM
dataprovider.gatDeleteConfigArnLoop.name=gatDeleteConfigArnLoop

dataprovider.generateArnlImportFileDataSource.type=tdm
dataprovider.generateArnlImportFileDataSource.context=System/ENM
dataprovider.generateArnlImportFileDataSource.name=generateArnlImportFileDat →
aSource

dataprovider.generateArnlErbsImportFile.type=tdm
dataprovider.generateArnlErbsImportFile.context=System/ENM
dataprovider.generateArnlErbsImportFile.name=generateArnlErbsImportFile

dataprovider.generateArnlRadioImportFile.type=tdm
dataprovider.generateArnlRadioImportFile.context=System/ENM
```

```
dataprovider.generateArnlRadioImportFile.name=generateArnlRadioImportFile

dataprovider.generateArnlMgwImportFile.type=tdm
dataprovider.generateArnlMgwImportFile.context=System/ENM
dataprovider.generateArnlMgwImportFile.name=generateArnlMgwImportFile

dataprovider.generateArnlRNCImportFile.type=tdm
dataprovider.generateArnlRNCImportFile.context=System/ENM
dataprovider.generateArnlRNCImportFile.name=generateArnlRNCImportFile

dataprovider.generateArnlSGSNImportFile.type=tdm
dataprovider.generateArnlSGSNImportFile.context=System/ENM
dataprovider.generateArnlSGSNImportFile.name=generateArnlSGSNImportFile

dataprovider.generateArnlRbsImportFile.type=tdm
dataprovider.generateArnlRbsImportFile.context=System/ENM
dataprovider.generateArnlRbsImportFile.name=generateArnlRbsImportFile

dataprovider.gatImportToLiveArnLoop.type=tdm
dataprovider.gatImportToLiveArnLoop.context=System/ENM
dataprovider.gatImportToLiveArnLoop.name=gatImportToLiveArnLoop

dataprovider.gatImportToNonLiveArnLoop.type=tdm
dataprovider.gatImportToNonLiveArnLoop.context=System/ENM
dataprovider.gatImportToNonLiveArnLoop.name=gatImportToNonLiveArnLoop

dataprovider.gatActionArnLoop.type=tdm
dataprovider.gatActionArnLoop.context=System/ENM
dataprovider.gatActionArnLoop.name=gatActionArnLoop

dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.type=tdm
dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.context=System/ENM
dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.name=gatCmeditGetAndSetCr →
eateDeleteArnLoop

dataprovider.gatFilterNonPersistent.type=tdm
dataprovider.gatFilterNonPersistent.context=System/ENM
dataprovider.gatFilterNonPersistent.name=gatFilterNonPersistent
#FGV
dataprovider.gatActionArnLoop .type=tdm
dataprovider.gatActionArnLoop .context=System/ENM
dataprovider.gatActionArnLoop .name=gatActionArnLoop

dataprovider.gatCmeditCreateDeleteArnLoop.type=tdm
dataprovider.gatCmeditCreateDeleteArnLoop.context=System/ENM
dataprovider.gatCmeditCreateDeleteArnLoop.name=gatCmeditCreateDeleteArnLoop

dataprovider.gatCmeditGetAndSetArnLoop.type=tdm
dataprovider.gatCmeditGetAndSetArnLoop.context=System/ENM
dataprovider.gatCmeditGetAndSetArnLoop.name=gatCmeditGetAndSetArnLoop

dataprovider.gatHeartBeatAndManualSyncArnLoop.type=tdm
dataprovider.gatHeartBeatAndManualSyncArnLoop.context=System/ENM
dataprovider.gatHeartBeatAndManualSyncArnLoop.name=gatHeartBeatAndManualSync →
ArnLoop

dataprovider.gatImportToLiveArnLoop.type=tdm
dataprovider.gatImportToLiveArnLoop.context=System/ENM
dataprovider.gatImportToLiveArnLoop.name=gatImportToLiveArnLoop

################################################################
#CM-NBI Specific Data
################################################################
dataprovider.gatConfigurationRootRequest.type=tdm
dataprovider.gatConfigurationRootRequest.context=System/ENM
dataprovider.gatConfigurationRootRequest.name=gatConfigurationRootRequest

dataprovider.gatCreateNonLiveConfig.type=tdm
dataprovider.gatCreateNonLiveConfig.context=System/ENM
dataprovider.gatCreateNonLiveConfig.name=gatCreateNonLiveConfig

dataprovider.gatCopyNeToNonLiveConfig.type=tdm
dataprovider.gatCopyNeToNonLiveConfig.context=System/ENM
dataprovider.gatCopyNeToNonLiveConfig.name=gatCopyNeToNonLiveConfig

dataprovider.gatDeleteNonLiveConfig.type=tdm
dataprovider.gatDeleteNonLiveConfig.context=System/ENM
dataprovider.gatDeleteNonLiveConfig.name=gatDeleteNonLiveConfig

dataprovider.gatActivateRevocation.type=tdm
```

```
dataprovider.gatActivateRevocation.context=System/ENM
dataprovider.gatActivateRevocation.name=gatActivateRevocation

dataprovider.gatImportRevocation.type=tdm
dataprovider.gatImportRevocation.context=System/ENM
dataprovider.gatImportRevocation.name=gatImportRevocation

dataprovider.gatBulkConfigRootRequest.type=tdm
dataprovider.gatBulkConfigRootRequest.context=System/ENM
dataprovider.gatBulkConfigRootRequest.name=gatBulkConfigRootRequest

dataprovider.gatImportXMLToNonLive.type=tdm
dataprovider.gatImportXMLToNonLive.context=System/ENM
dataprovider.gatImportXMLToNonLive.name=gatImportXMLToNonLive

dataprovider.gatImportZippedXMLToNonLive.type=tdm
dataprovider.gatImportZippedXMLToNonLive.context=System/ENM
dataprovider.gatImportZippedXMLToNonLive.name=gatImportZippedXMLToNonLive

dataprovider.gatImportDynamicToNonLive.type=tdm
dataprovider.gatImportDynamicToNonLive.context=System/ENM
dataprovider.gatImportDynamicToNonLive.name=gatImportDynamicToNonLive

dataprovider.gatImportDynamicToLive.type=tdm
dataprovider.gatImportDynamicToLive.context=System/ENM
dataprovider.gatImportDynamicToLive.name=gatImportDynamicToLive

dataprovider.generateImportFileXml.type=tdm
dataprovider.generateImportFileXml.context=System/ENM
dataprovider.generateImportFileXml.name=generateImportFileXml

dataprovider.generateImportFileZip.type=tdm
dataprovider.generateImportFileZip.context=System/ENM
dataprovider.generateImportFileZip.name=generateImportFileZip

dataprovider.generateImportFileDynamic.type=tdm
dataprovider.generateImportFileDynamic.context=System/ENM
dataprovider.generateImportFileDynamic.name=generateImportFileDynamic

dataprovider.gatUser.type=tdm
dataprovider.gatUser.context=System/ENM
dataprovider.gatUser.name=gatUser

dataprovider.generateERBSImport3GPPFile.type=tdm
dataprovider.generateERBSImport3GPPFile.context=System/ENM
dataprovider.generateERBSImport3GPPFile.name=generateERBSImport3GPPFile

dataprovider.generateERBSImportDynamicFile.type=tdm
dataprovider.generateERBSImportDynamicFile.context=System/ENM
dataprovider.generateERBSImportDynamicFile.name=generateERBSImportDynamicFil →
e

dataprovider.generateERBSZippedImportXmlFile.type=tdm
dataprovider.generateERBSZippedImportXmlFile.context=System/ENM
dataprovider.generateERBSZippedImportXmlFile.name=generateERBSZippedImportXm →
lFile

dataprovider.gatFilterPersistent.type=tdm
dataprovider.gatFilterPersistent.context=System/ENM
dataprovider.gatFilterPersistent.name=gatCopyNeToNonLiveConfig

dataprovider.gatImportXmlToLive.type=tdm
dataprovider.gatImportXmlToLive.context=System/ENM
dataprovider.gatImportXmlToLive.name=gatImportXmlToLive

dataprovider.generateRevokeImport.type=tdm
dataprovider.generateRevokeImport.context=System/ENM
dataprovider.generateRevokeImport.name=generateRevokeImport

dataprovider.gatImportDynamicAndRevokeFileToLive.type=tdm
dataprovider.gatImportDynamicAndRevokeFileToLive.context=System/ENM
dataprovider.gatImportDynamicAndRevokeFileToLive.name=gatImportDynamicAndRev →
okeFileToLive

arn.import.config.nonlive=importConfig
arn.import.config.live=Live
arn.import.dynamic.filetype=dynamic
arn.import.3gpp.filetype=3GPP
modifier.update=update
```

The table below shows a summary of the CM-dependent TDM data sources with a short description of their usage:

TDM Data Source	Description
gatActionArnLoop	Adding test data for: GAT-CM: Execute a modeled action on a Managed Object This TDM data source contains information for the test where an attribute is deleted and then recreated. These steps are then verified for accuracy.
gatCmeditGetAndSetCreateDeleteArnLoop	Adding test data for: GAT-CM: Get and Set Attributes on a Managed Object on NE; GAT-CM: Create and Delete Managed Objects on NE This TDM data source contains the parameters for the test where a node MO is gotten, deleted and then a new MO is created in its place and then this MO is then deleted.
gatConfigArnLoop	Adding test data for: GAT-CM: Create a Non Live Configuration and List configurations; GAT-CM: Copy NE to a non live configuration, Modify NE data, Activate configuration and Undo Changes; GAT-CM: Delete a Non Live Configuration This TDM data source has the parameters for the test where a non live configuration is created, a node is copied to it, this node is then edited and the changes are activated, undone and a file is imported to the non live config. When the test is done the non live config is deleted.
gatHeartbeatAndManualSyncArnLoop	Adding test data for: GAT-ADD: Start CM Supervision and Sync a Node This TDM data source contains parameter information about the node and the node type. This information is used in the test where the node is unsynced, synced, the neType is described, the node information is gotten and the node is exported.
verifyNonPersistedSetAndGetArnLoop	Adding test data for: GAT-CM: Get and Set Attributes on a Managed Object on NE; Non persisted This TDM data source contains information to be used in editing a non persisted attribute on a node. This change is then checked and verified.
generateArnlImportFileDataSource	Adding test data for: GAT-CM: Import File into Live Configuration. File Generation test data This TDM data source lists the data sources used in the import tests. In the generateImportFileDatasource column it references the data sources that are to create the import files. The generateImportFileType specifies the type of file (Dynamic or 3GPP) to be created and the generateFileNameToBeCreated column specifies the name of the import file to be made.
gatImportArnLoop	Import Test Case test data This TDM data source obtains the import files created with the generateArnlImportFileDataSource and imports them into ENM. The changes made by the import job are then checked and verified in order to ensure that they are correct.
generateArnl<NodeType>ImportFile	Adding test data for: GAT-CM: Import File into Live Configuration This TDM data source is needed for each configured node type, represented by <NodeType> on file name. It specifies the information that is used to create an import file for an NodeType node. This file must be referenced in the generateArnlImportFileDataSource. The modifier action to be performed, the MO ID, the MO to be

TDM Data Source	Description
	edited, the attributes to be edited and the FDN of the node must be provided.
gatCmeditGetAndSetArnLoop	Adding test data for: GAT-CM: Get and Set Attributes on a Managed Object on NE This TDM data source contains information for the test where a node is got and then information for it is set. These steps are then verified for accuracy.
gatCmeditCreateAndDeleteArnLoop	Adding test data for: GAT-CM: Create and Delete Managed Objects on NE This TDM data source contains information for the test where an FDN is deleted, created and then deleted again. The test steps are then verified.
gatCreateListConfigArnLoop	Adding test data for: GAT-CM: Create a Non Live Configuration and List configurations This TDM data source contains information for the test where non live configuration is created and then it is listed in order to ensure it exists. These steps are then verified for accuracy.
gatCopyModifyActivateUndoConfigArnLoop	Adding test data for: Copy NE to a non live configuration, Modify NE data, Activate configuration and Undo Changes This TDM data source contains information for the test where a node is copied to a the non live config created in the previous test, information is then set and the changes are then activated. After this the history of the changes are checked and then undone. These steps are then verified for accuracy
gatDeleteConfigArnLoop	Adding test data for: GAT-CM: Delete a Non Live Configuration This TDM data source contains information for the test where the non live configuration created in the gatCreateListConfigArnLoop test is deleted. These steps are then verified for accuracy.

3. Additional configuration is needed for each different node present in `NodeData`, additional detail can be seen in Appendix A. In general, the following manual updates are needed:

   — In `/root/agat/tdm.properties` file, add node type to the `filter.node` property and the new node credentials:

```
# CM Specific Data
filter.node=RadioNode;ERBS;RBS;RNC;SGSN-MME;MGW

# Node Data - RadioNode 1
arn.nodes.radio.node1=<network element name>
arn.nodes.radio.node1.ossprefix=SubNetwork=G2RBS\\,ManagedElement=${arn
.nodes.radio.node1}

# Node Data - ERBS 1
arn.nodes.erbs.node1=<network element name>
arn.nodes.erbs.node1.ossprefix=MeContext=${arn.nodes.erbs.node1}

# Node Data - RBS 1
arn.nodes.rbs.node1=<network element name>
arn.nodes.rbs.node1.ossprefix=MeContext=${arn.nodes.rbs.node1}

# Node Data - RNC 1
arn.nodes.rnc.node1=<network element name>
arn.nodes.rnc.node1.ossprefix=MeContext=${arn.nodes.rnc.node1}

# Node Data - SGSN 1
arn.nodes.sgsn.node1=<network element name>
arn.nodes.sgsn.node1.ossprefix=SubNetwork=NETSimW\\,ManagedElement=${ar
n.nodes.sgsn.nodes1}

# Node Data - MGW 1
arn.nodes.mgw.node1=<network element name>
```

```
arn.nodes.mgw.node1.ossprefix=SubNetwork=NETSimW\\,ManagedElement=${arn
.nodes.mgw.node1}
```

- Add an additional row in each of the TDM data sources in the **cm-cli** for that specific node.
- For each node type, create a new `generateArnl<node>ImportFile` data source under **cm-cli**.

5.5 Additional Configuration for Agat_PM_CTUM Suite

The following conditions must be met in order to run Agat_PM_CTUM suite:

- Before the start of the test, the CTUM subscription must be in an inactive state.
- The output mode in the subscription must be `File Only`.

5.6 Additional Configuration for Agat_CM_DeleteNode Suite

- The node configured for `Agat_CM_DeleteNode` suite in the `NodeData.sample` data source should be a test node as the node will be deleted as a part of the Delete Node testcase.
- The node configured for `Agat_CM_DeleteNode` suite in the `NodeData.sample` data source should not be used for any other suites.
- If a dummy node is being used to test this suite, then it is expected that the Start CM Supervision and Sync a Node test case will be in broken state as a dummy node cannot be synced.

6 Execute Tests

> Verify prerequisites before executing the testware.

The following prerequisite files exist with all the correct configuration information such as Network ID, user, password:

— Schedule File (`agat_schedule_yyyymmddhhmmss.xml`)

— Host Properties File (`host.properties.json`)

— Test Properties File (`tdm.properties`)

— Data Sources

— Trigger jar (`taf-trigger-jar-<version>-jar-with-dependencies.jar`)

Note: If any of the above files are changed, restart the slave container with the following command:

```
docker restart `docker ps -q -f 'NAME=taf_te_slave'`
```

There are two different ways to execute AGAT testware:

— — Execute AGAT from the Command Line

— — Execute AGAT Using AAT

Note: One of these options can be followed.

It is not required to complete both.

6.1 Execute AGAT from the Command Line

Prerequisites

AAT is installed and the AGAT service is registered in the AAT GUI.

Steps

1. Navigate to the `/root/agat/` directory:

```
[root@localhost~]# cd /root/agat/
```

2. Run the following command to execute the testware:

```
[root@localhost~ agat]# java -jar <path to TAF trigger jar> taf.version=<taf
 version> taf.te.address=<te host address> taf.test.properties=tdm.propertie
s taf.host.properties=host.properties.json taf.schedule=agat_schedule_yyyymm
ddhhmmss.xml taf.te.user=< te username> taf.te.password=<te password>
```

All parameters listed are mandatory and must be included in the execution of the command when executing the test ware.

Parameter	Description
`taf.host.properties`	File containing the host properties (IP addresses, users, and so on)
`taf.schedule`	File containing the schedule XML specifying the testware to be run and execution order.
`taf.test.properties`	File containing any additional properties need by the testware.
`taf.te.address`	The address of the host which the TE Docker containers are running.
`taf.version`	The TAF version can be obtained from the `MINIMUM_TAF_VERSION.txt` file in `/root/.m2/repository` on the AGAT server.
`<path to TAF trigger jar>`	The TAF trigger jar file is required to execute the AGAT tests. This file can be obtained from Test Executor package. The user needs to copy this file to the `/root/agat` directory.
`taf.te.user`	The user added while configuring LDAP user during installation of TE.
`taf.te.password`	The password of the `taf.te.user` added while configuring LDAP user during installation of TE.

6.2 Execute AGAT using AAT

Prerequisites

AAT is installed and the AGAT service is registered in the AAT GUI.

Steps

1. Open a web browser and go to the server where the AAT Core is installed

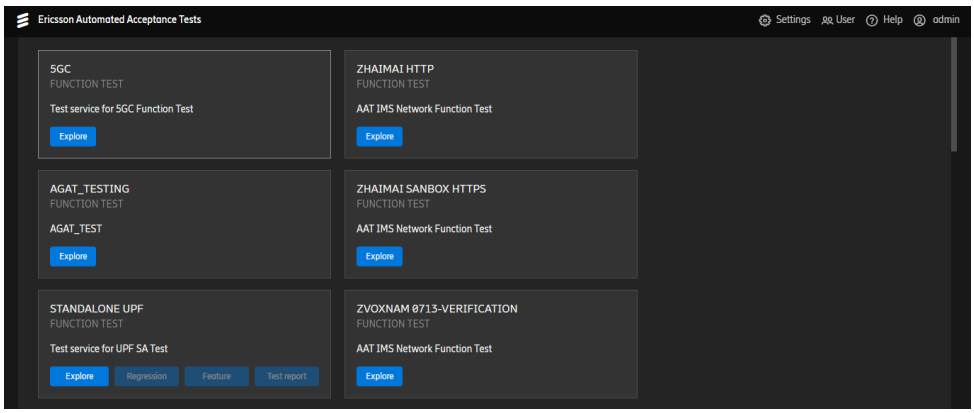
   `(<IPaddress>:<port>).`

   Example:

   ```
   https://<AAT Installed Server IP>:<AAT UI port>/login
   ```

2. Log with the valid username and password.

   After logging on, the AAT main page is displayed. (This page can be logically divided into two parts, the AAT Toolbar and the AAT Service list).



6.2.1 Configure Test Executor

Complete the parameter details to configure Test Executor.

Steps

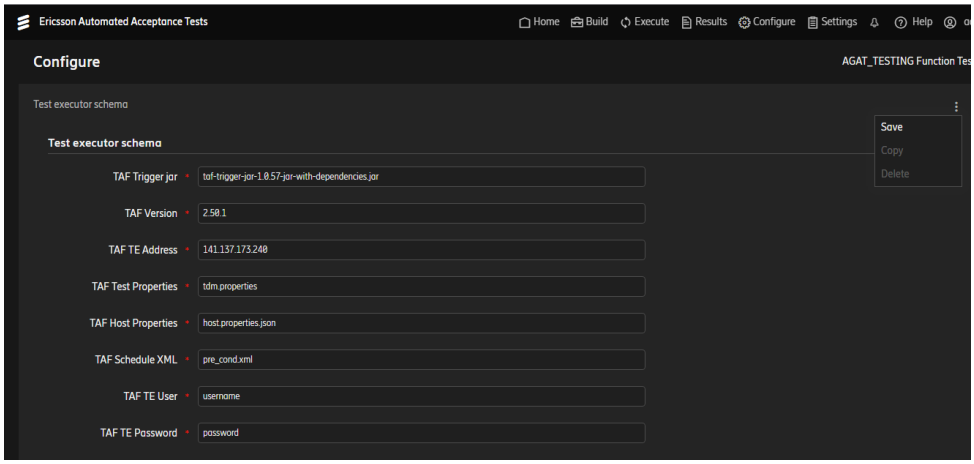
1. On the Configure page, fill in the fields under Test Executor Schema.

   Use the following table as a guide.

Parameter	Description
taf.host.properties	File containing the host properties (IP addresses, users, etc)
taf.schedule	File containing the schedule XML specifying the testware to be run and execution order
taf.test.properties	File containing any additional properties need by the testware.
taf.te.address	The address of the host which the TE Docker containers are running.
taf.version	The TAF version can be obtained from the `MINIMUM_TAF_VERSION.txt` file located at `/root/.m2/repository` on the AGAT server.
<path to TAF trigger jar>	The TAF trigger jar file is required to execute the AGAT tests. This file can be obtained from TestExecutor package. The user needs to copy this file to the `/root/agat` directory.
taf.te.user	The user added while configuring LDAP user during installation of TE

Parameter	Description
taf.te.password	The password of the `taf.te.user` added while configuring LDAP user during installation of TE.

2. Click Save to store your configuration details.

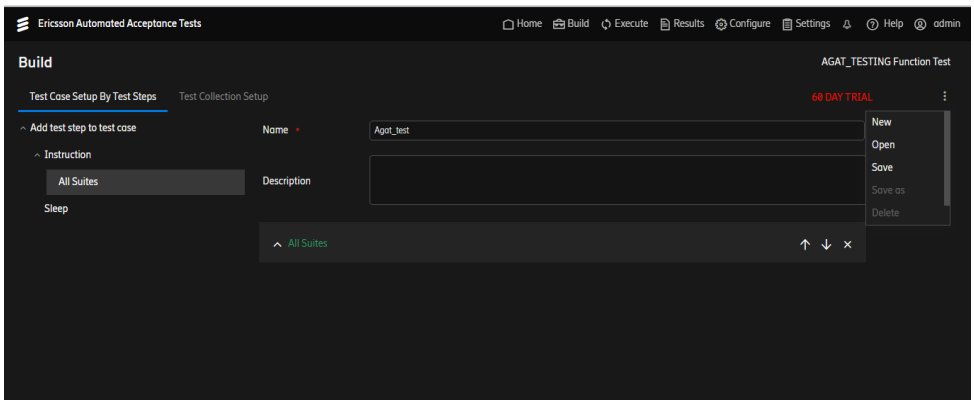


6.2.2 Build a Test Case

Use the options on the Build tab to create a test case.

Steps

1. Click New to define a new test case.
2. Enter a Name and Description for the new test case.
3. select All Suites.This adds a test step to the test case.
4. Save the defined test case so that it is stored for use in the execution phase.



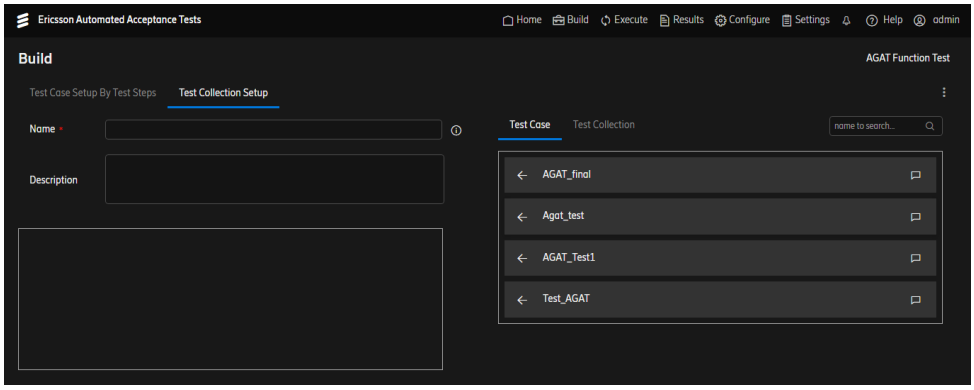
6.2.3 Build a Test Collection

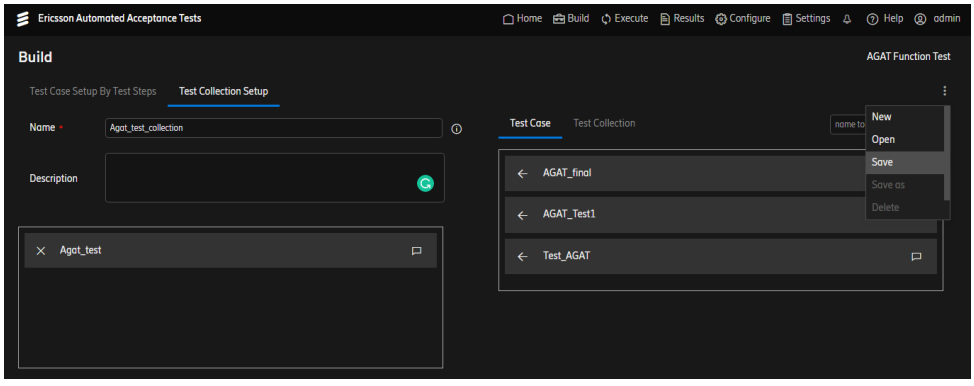
Use the options on the Build tab to create a test collection

Steps

1. Click **New** to define a new test case collection for the test case collection library
2. Enter a Name and Description for the test collection. case.
3. Select **only one Test case** from list of Test cases to add to the Test Collection Setup.

   Note: AGAT supports one Test case for Test collection Setup.

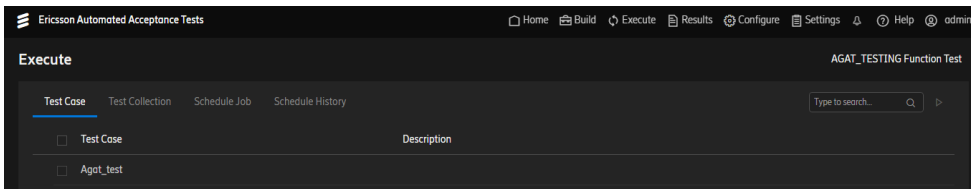




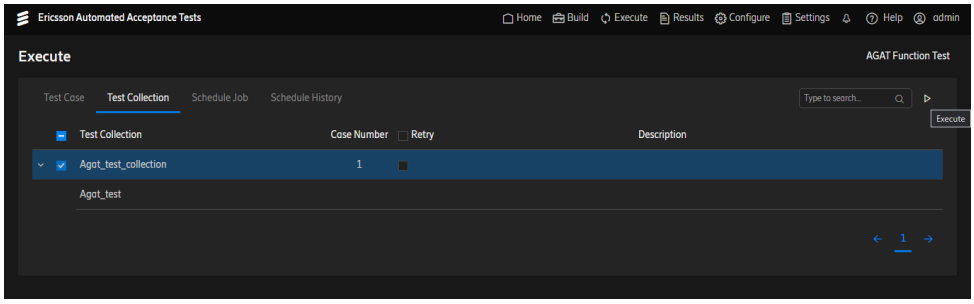
6.2.4 Execute a Test Case/Test Collection

Steps

1. On the Execute tab, click `Execute` to start the test case



2. On the Execute tab, select **Test Collection** tab, select **Test Collection** and click **Execute** to start the test collection.



3. There are other options under **Execute** section

   — **Schedule Job**: Schedule a Test Collection to run Once/Daily at a Scheduled Date and Time.

   — **Schedule History**: List of Schedule Jobs that ran and their status.

6.2.5 Execution Progress and Result

Results of the test case are displayed on the Results tab according to the following headings:

Steps

— Latest Results

  The results of the most recent executions.

— Passed, Failed

  Count of the passed/failed test run execution.

— Start Time

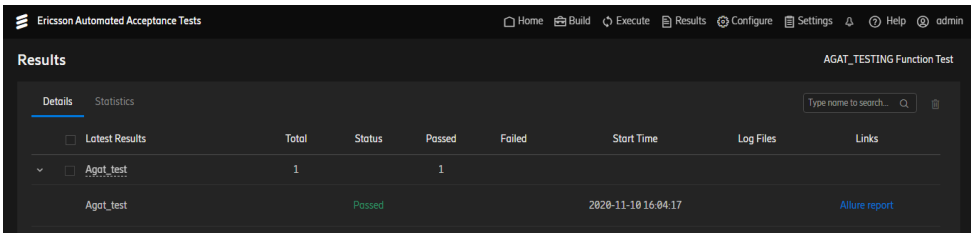
  Execution start time

— Links

  Links to the Allure report that provides a detailed breakdown of the test run.

— Status

Status of the total test run execution (passed/failed/skipped).



6.2.6 Troubleshooting

To check the status of the testware running:

Steps

1. check the logs of the AAT container:

   ```
   docker logs <aat_container ID>
   ```

   Above command request provides the relevant error or result:

   Example

   ```
   2020-11-10 10:30:59.084583 : POST Request with JobID- cbf321c
   c-4990-45f2-92f2-523b197061ad
   Test execution command is: java -jar taf-trigger-jar-1.0.57-j
   ar-with-dependencies.jar taf.version=2.50.1 taf.te.address=14
   1.137.173.240 taf.test.properties=tdm.properties taf.host.pro
   perties=host.properties.json taf.schedule=pre_cond.xml taf.te
   .user=username taf.te.password=****
   ============================================================
   ==================
   TE process has been initiated. Waiting for tests to finish...
   ============================================================
   ==================
   New test master job created: TEST_SCHEDULER #7:https://141.13
   7.173.240:8443/jenkins/job/TEST_SCHEDULER/7/consoleFull

   New test execution job created: AGAT_PreCond:https://141.137.
   173.240:8443/jenkins/job/TEST_EXECUTOR/85/consoleFull

   TEST_SCHEDULER #7 has finished - SUCCESS
   AGAT_PreCond has finished - SUCCESS
   ============================================================
   ==================
   Secure Allure Logs: https://141.137.173.240:443/3d6ce3f5-23a9
   -4bca-8c4d-c19c3d4accca/
   Allure Logs: http://141.137.173.240:8088/3d6ce3f5-23a9-4bca-8
   c4d-c19c3d4accca/
   ```

7 Use Allure Test Reports

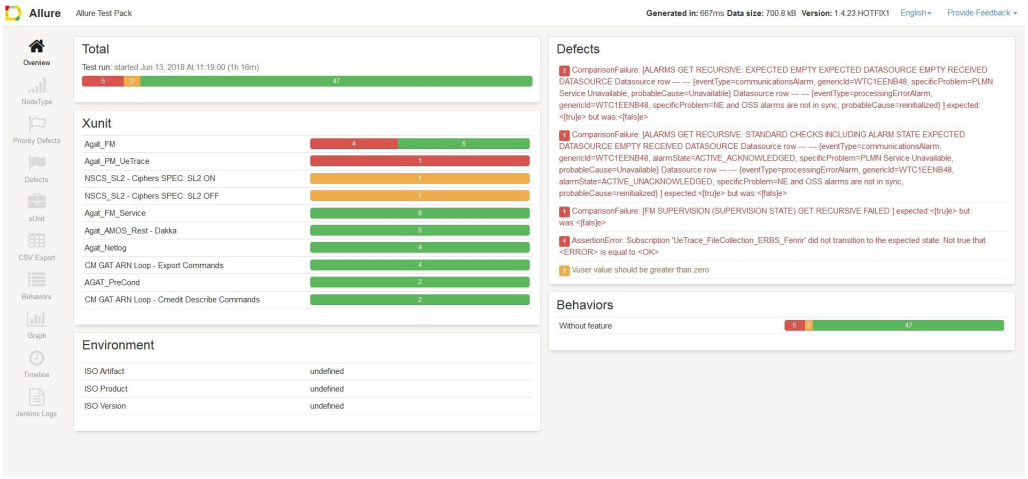
> Allure is a 3PP framework for creating an HTML test report. An Allure report is generated after a test execution has completed.

In the distribution provided by TE, there are additional plug-ins added to this report.

For more information on Allure and its standard HTML report features, refer to Allure documentation: `https://docs.qameta.io/allure/`.

A **Sign in** dialog box is displayed when you try to open an Allure report. Enter your valid LDAP username and password to sign in and view the report.

Sample Allure Report



Retrieving Allure Reports

At the end of the test execution, the URL for the Allure report is displayed on the console in one of the following formats:

— Secure Allure Logs: `https://<Host IP Address>:443/<Job Execution Id>/`

— Allure Logs: `http://<Host IP Address>:8088/<Job Execution Id>/`

Copy and paste the link into the browser to view the log.

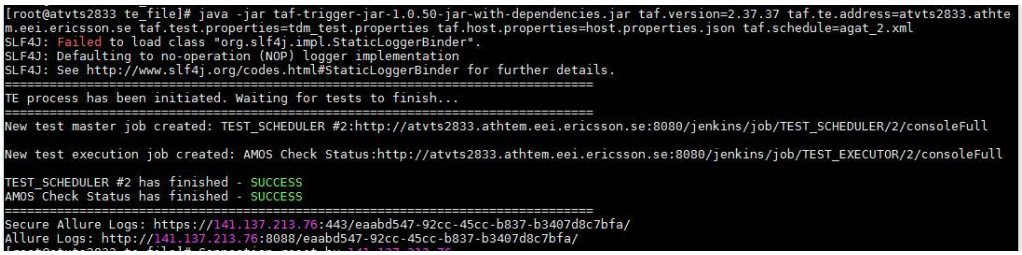


Figure 13 Sample Console Log Output

Allure Report Tabs

Overview — Lists an overall test execution summary, highlights of defects, environment variables, behaviors, and test suite execution details.

Note: The **Overview** tab displays only 10 suites. However, the execution results of all the suites can be checked under **xUnit** or **Defects**.

Defects — Lists all failing tests both failed or broken. It is possible to select all items and drill down for more information.

xUnit — Lists all tests in a report based on their executed result. Click the test suite to display the executed tests. Filter tabs allow you to filter on the results of the tests executed.

Clicking the test case in the right panel drills into the test and the actions performed.

Behaviors — Shows a breakdown of each test case with feature ID connected to a customer requirement. This is recommended for internal use.

Graph — Displays charts on test case status, test case severity, and test duration.

Timeline — Shows the time that the test was executed and its duration across the entire execution of tests.

Additional Ericsson Plugins

Note: CSV Export and Jenkins Logs are recommended tabs that can be used to analyze tests. Other tabs are optional views.

CSV Export — Allows users to download the HTML data into a CSV file.

Jenkins Logs — Shows all links to all Jenkins logs for each test execution.

Priority Defects

Groups all broken and failing tests by test priority, for example: blocker, normal, and minor.

Node Type

Shows all test cases based on a node type.

Team Name

Shows each test case per team name.

KPIs

Displays KPI data from the test cases.

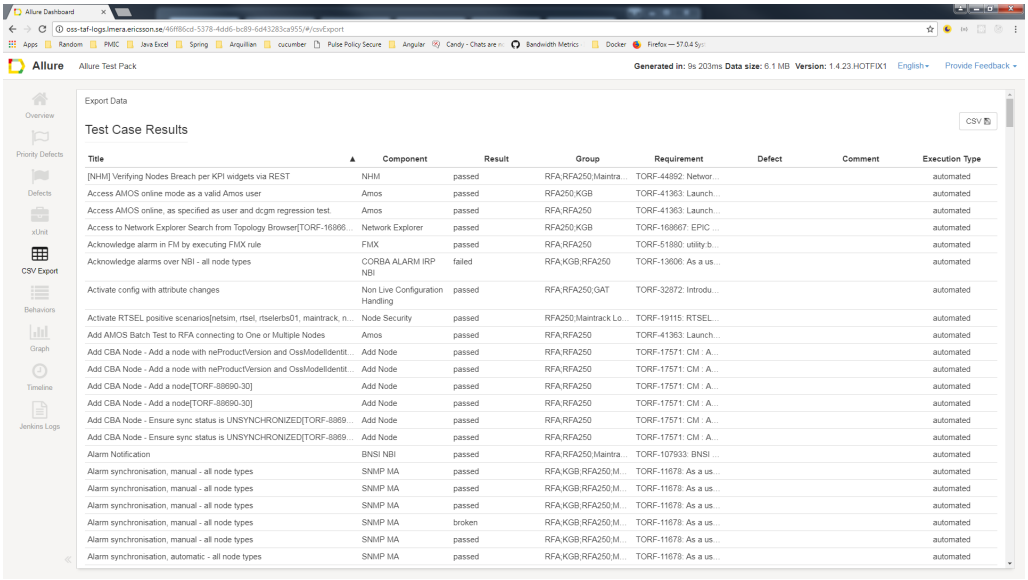


Figure 14 CSV Export Tab Example

8 Appendix A - Add / Configure Datasources for CM

> This appendix describes how to add and update test data from external sources which includes TDM data sources stored remotely.

Start CM Supervision and Sync a Node

The configuration is done in `/root/agat/tdm.properties` file and `gatHeartbeatAndManualSyncArnLoop` data source.

Check if the properties are already configured in `/root/agat/tdm.properties`. If not, add or edit properties as required:

```
dataprovider.gatHeartbeatAndManualSyncArnLoop.type= tdm
dataprovider.gatHeartbeatAndManualSyncArnLoop.id=5c6fe4e836d1a00001253eb8
```

Each node represents a new row in `gatHeartbeatAndManualSyncArnLoop`, as in the example. The `dataprovider.gatHeartbeatAndManualSyncArnLoop.id` property must be pointing to `gatHeartbeatAndManualSyncArnLoop` data source ID.

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<neType>,nodeType
${filter.node},"Start CM Supervision and Sync a Node - ERBS",${arn.nodes.erbs.no →
de1},${arn.nodes.erbs.netype},${arn.nodes.erbs.netype}
${filter.node},"Start CM Supervision and Sync a Node - Radionode",${arn.nodes.ra →
dio.node1},${arn.nodes.radio.netype},${arn.nodes.radio.netype}
${filter.node},"Start CM Supervision and Sync a Node - RBS",${arn.nodes.rbs.node →
1},${arn.nodes.rbs.netype},${arn.nodes.rbs.netype}
${filter.node},"Start CM Supervision and Sync a Node - RNC",${arn.nodes.rnc.node →
1},${arn.nodes.rnc.netype},${arn.nodes.rnc.nodeType}
${filter.node},"Start CM Supervision and Sync a Node - MGW",${arn.nodes.mgw.node →
1},${arn.nodes.mgw.netype},${arn.nodes.mgw.nodeType}
${filter.node},"Start CM Supervision and Sync a Node - SGSN",${arn.nodes.sgsn.no →
de1},${arn.nodes.sgsn.netype},${arn.nodes.sgsn.nodeType}
```

To add a new node, add a new row to `gatHeartbeatAndManualSyncArnLoop`, for example:

```
${filter.node},"GAT-ADD: Start CM Supervision and Sync a Node - MGW",${arn.nodes →
.mgw.node1},${arn.nodes.mgw.netype},${arn.nodes.mgw.netype}
```

Some of the values in `gatHeartbeatAndManualSyncArnLoop` represent a property in `/root/agat/tdm.properties`, they are:

- `filter.node`
- `arn.nodes.<nodeType>.node<number>`
- `arn.nodes.<nodeType>.netype`

Configure these properties in `/root/agat/tdm.properties`. See `/root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties` template and Appendix B for examples.

Get and Set Attributes on a Managed Object on NE

The configuration is done in `/root/agat/tdm.properties` file and `gatCmeditGetAndSetCreateDeleteArnLoop` data source.

Check if the following properties are already configured in `/root/agat/tdm.properties`. If not, add or edit properties as required.

```
dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.type= tdm
dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.id=5c6fe47536d1a00001253e70
```

The `dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.id` property must be pointing to `gatCmeditGetAndSetCreateDeleteArnLoop` data source ID.

Each node represents a new row in `gatCmeditGetAndSetCreateDeleteArnLoop`, as in the example:

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<MO_to_use>,<attribute_ →
to_use>,<new_value>,<another_new_value>,<FDN_of_MO_to_use>,<List_of_MO_attribute →
s_and_values>,nodeType
${filter.node},"Get and Set Attributes on a Managed Object on NE; GAT-CM: Create →
 and Delete Managed Objects on NE - ERBS",${arn.nodes.erbs.node1},${arn.get.set. →
erbs.mo},${arn.get.set.erbs.mo.attribute},${arn.get.set.erbs.mo.attribute.value} →
,${arn.get.set.erbs.mo.attribute.new.value},${arn.get.set.erbs.create.fdn},${arn →
.get.set.erbs.create.fdn.attributes},${arn.nodes.erbs.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE; GAT-CM: Create →
 and Delete Managed Objects on NE - Radionode",${arn.nodes.radio.node1},${arn.ge →
t.set.radio.mo},${arn.get.set.radio.mo.attribute},${arn.get.set.radio.mo.attribu →
te.value},${arn.get.set.radio.mo.attribute.new.value},${arn.get.set.radio.create →
.fdn},${arn.get.set.radio.create.fdn.attributes},${arn.nodes.radio.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE; GAT-CM: Create →
 and Delete Managed Objects on NE - RBS",${arn.nodes.rbs.node1},${arn.get.set.rb →
s.mo},${arn.get.set.rbs.mo.attribute},${arn.get.set.rbs.mo.attribute.value},${ar →
n.get.set.rbs.mo.attribute.new.value},${arn.get.set.rbs.create.fdn},${arn.get.se →
t.rbs.create.fdn.attributes},${arn.nodes.rbs.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE; GAT-CM: Create →
 and Delete Managed Objects on NE - RNC",${arn.nodes.rnc.node1},${arn.get.set.rn →
c.mo},${arn.get.set.rnc.mo.attribute},${arn.get.set.rnc.mo.attribute.value},${ar →
n.get.set.rnc.mo.attribute.new.value},${arn.get.set.rnc.create.fdn},${arn.get.se →
t.rnc.create.fdn.attributes},${arn.nodes.rnc.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE; GAT-CM: Create →
 and Delete Managed Objects on NE - SGSN",${arn.nodes.sgsn.node1},${arn.get.set. →
sgsn.mo},${arn.get.set.sgsn.mo.attribute},${arn.get.set.sgsn.mo.attribute.value} →
,${arn.get.set.sgsn.mo.attribute.new.value},${arn.get.set.sgsn.create.fdn},${arn →
.get.set.sgsn.create.fdn.attributes},${arn.nodes.sgsn.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE; GAT-CM: Create →
 and Delete Managed Objects on NE - MGW",${arn.nodes.mgw.node1},${arn.get.set.mg →
w.mo},${arn.get.set.mgw.mo.attribute},${arn.get.set.mgw.mo.attribute.value},${ar →
n.get.set.mgw.mo.attribute.new.value},${arn.get.set.mgw.create.fdn},${arn.get.se →
t.mgw.create.fdn.attributes},${arn.nodes.mgw.netype}
```

To add a new node, add a new row to `gatCmeditGetAndSetCreateDeleteArnLoop`, for example:

```
${filter.node},"Get and Set Attributes on a Managed Object on NE; GAT-CM: Create →
 and Delete Managed Objects on NE - MGW",${arn.nodes.mgw.node1},${arn.get.set.mg →
w.mo},${arn.get.set.mgw.mo.attribute},${arn.get.set.mgw.mo.attribute.value},${ar →
n.get.set.mgw.mo.attribute.new.value},${arn.get.set.mgw.create.fdn},${arn.get.se →
t.mgw.create.fdn.attributes},${arn.nodes.mgw.netype}
```

Some of the values in `gatCmeditGetAndSetCreateDeleteArnLoop` represent a property in `/root/agat/tdm.properties`, they are:

- filter.node
- arn.nodes.<nodeType>.node<number>
- arn.get.set.<nodeType>.mo
- arn.get.set.<nodeType>.mo.attribute
- arn.get.set.<nodeType>.mo.attribute.value
- arn.get.set.<nodeType>.mo.attribute.new.value
- arn.get.set.<nodeType>.create.fdn
- arn.get.set.<nodeType>.create.fdn.attributes
- arn.nodes.<nodeType>.netype

Configure these properties in `/root/agat/tdm.properties`. See `/root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties` template and Appendix B for examples.

Import File into Live Configuration

Import is a special case where each node type must have an import operations data source, which must be referenced in `/root/agat/tdm.properties` and added to a main import data source called `generateArnlImportFileDataSource`. It allows multiple import operations for each node type.

Check if the below properties referencing the `generateArnlImportFileDataSource` data source are already configured in `/root/agat/tdm.properties`. If not, add or edit as required.

```
dataprovider.generateArnlImportFileDataSource.type=tdm
dataprovider.generateArnlImportFileDataSource.id=5c6fe53436d1a00001253ede
```

For each node type, create an import operations data source, that must be named following this pattern `generateArnl<Nodetype>ImportFile`, inside. There are two template data sources, `generateArnlRadioImportFile` and `generateArnlErbsImportFile`, that can be copied to generate a new data source for a new node type if necessary.

This file contains import operations to be done by the node type. Each import operation is a row in the data source. Below is an example of `generateArnlErbsImportFile`:

```
modifier,fdn,mo,mo_id,attributes_to_modify
${modifier.update},${arn.import.erbs.dynamic.fdn},${arn.import.erbs.mo.to.change
},"1","${arn.import.erbs.mo.to.change}:""${arn.import.erbs.mo.attribute.value}""
"
```

For each import operations file created, like `generateArnlRadioImportFile` and `generateArnlErbsImportFile`, it is necessary to add a row in `generateArnlImportFileDataSource` data source.

For example, having six import operations data sources: `generateArnlRadioImportFile`, `generateArnlErbsImportFile`, `generateArnlMgwImportFile`, `generateArnlRbsImportFile`, `generateArnlRNCImportFile`, `generateArnlSGSNImportFile`; the `generateArnlImportFileDataSource` must contains six rows:

```
generateImportFileDatasource,generateImportFileType,generateImportFileNameToBeCr
eated,nodeType
"generateArnlErbsImportFile",dynamic,${arn.import.erbs.dynamic.filename},${arn.n
odes.erbs.netype}
"generateArnlRadioImportFile",3GPP,${arn.import.radio.3gpp.filename},${arn.nodes
.radio.netype}
"generateArnlRbsImportFile",dynamic,${arn.import.rbs.dynamic.filename},${arn.nod
es.rbs.netype}
"generateArnlRNCImportFile",dynamic,${arn.import.rnc.dynamic.filename},${arn.nod
es.rnc.nodeType}
"generateArnlSGSNImportFile",dynamic,${arn.import.sgsn.dynamic.filename},${arn.n
odes.sgsn.nodeType}
"generateArnlMgwImportFile",dynamic,${arn.import.mgw.dynamic.filename},${arn.nod
es.mgw.nodeType}
```

Note: The names of the `generateArnl<NodeType>ImportFile` in the `generateImportFileDatasource` column must have exactly the same name as the file listed in the `tdm.properties` file.

Also, each import operations file must be referenced in `/root/agat/tdm.properties`. Using the previous example, it is configured as follows:

```
dataprovider.generateArnlErbsImportFile.type=tdm
dataprovider.generateArnlErbsImportFile.id=5c6fe51536d1a00001253ed8

dataprovider.generateArnlRadioImportFile.type=tdm
dataprovider.generateArnlRadioImportFile.id=5c6fe56036d1a00001253ef4

dataprovider.generateArnlRbsImportFile.type=tdm
dataprovider.generateArnlRbsImportFile.id=5c6fe57a36d1a00001253efc

dataprovider.generateArnlRNCImportFile.type=tdm
dataprovider.generateArnlRNCImportFile.id=5c6fe58e36d1a00001253f02

dataprovider.generateArnlSGSNImportFile.type=tdm
dataprovider.generateArnlSGSNImportFile.id=5c6fe5a136d1a00001253f08

dataprovider.generateArnlMgwImportFile.type=tdm
dataprovider.generateArnlMgwImportFile.id=5c6fe54b36d1a00001253eee
```

Some of the values in the presented data sources represent a property in `/root/agat/tdm.properties`, they are:

— In `generateArnlImportFileDataSource`

- arn.import.<nodeType>.dynamic.filename or arn.import.<nodeType>.3gpp.filename
- generateArnl<nodeType>ImportFile: part of dataprovider.generateArnl<nodeType>ImportFile.* properties
- arn.nodes.<nodeType>.nodeType

— In generateArnl<Nodetype>ImportFile

- modifier.update
- arn.import.<nodeType>.mo.attribute.value
- arn.import.<nodeType>.mo.to.change
- arn.import.<nodeType>.mo.to.change
- arn.import.<nodeType>.dynamic.fdn or arn.import.<nodeType>.3gpp.fdn

Configure these properties in /root/agat/tdm.properties. See /root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties template and Appendix B for examples.

Execute a Modeled Action on a Managed Object

The configuration is done in /root/agat/tdm.properties file and gatActionArnLoop data source.

Check if the following properties are already configured in /root/agat/tdm.properties. If not, add or edit as required:

```
dataprovider.gatActionArnLoop.type=tdm
dataprovider.gatActionArnLoop.id=5c4af2f536d1a000012539a3
```

The dataprovider.gatActionArnLoop.id property must be pointing to the gatActionArnLoop data source ID.

Each node represents a new row in gatActionArnLoop, as in the example:

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<checkDeletedCmd>,<FDN> →
,<createAction>,<createAttributes>,<deleteAction>,<deleteAttributes>,<getFDN>,<n →
oneFound>,<oneFound>,nodeType
${filter.node},"Execute a modeled action on a Managed Object - ERBS",${arn.nodes →
.erbs.node1},${arn.action.erbs.check.delete.command},${arn.action.erbs.fdn},${ar →
n.action.erbs.create.action},${arn.action.erbs.create.attributes},${arn.action.e →
rbs.delete.action},${arn.action.erbs.delete.attributes},${arn.action.erbs.get.fd →
n},${arn.action.erbs.none.found},${arn.action.erbs.one.found},${arn.nodes.erbs.n →
etype}
${filter.node},"Execute a modeled action on a Managed Object - Radionode",${arn. →
nodes.radio.node1},${arn.action.radio.check.delete.command},${arn.action.radio.f →
dn},${arn.action.radio.create.action},${arn.action.radio.attributes},${arn.actio →
n.radio.delete.action},${arn.action.radio.attributes},${arn.action.radio.get.fdn →
},${arn.action.radio.none.found},${arn.action.radio.one.found},${arn.nodes.radio →
.netype}
${filter.node},"Execute a modeled action on a Managed Object - RBS",${arn.nodes. →
```

```
rbs.node1},${arn.action.rbs.check.delete.command},${arn.action.rbs.fdn},${arn.ac
tion.rbs.create.action},${arn.action.rbs.create.attributes},${arn.action.rbs.del
ete.action},${arn.action.rbs.delete.attributes},${arn.action.rbs.get.fdn},${arn.
action.rbs.none.found},${arn.action.rbs.one.found},${arn.nodes.rbs.netype}
${filter.node},"Execute a modeled action on a Managed Object - SGSN",${arn.nodes
.sgsn.node1},${arn.action.sgsn.check.delete.command},${arn.action.sgsn.fdn},${ar
n.action.sgsn.create.action},${arn.action.sgsn.create.attributes},${arn.action.s
gsn.delete.action},${arn.action.sgsn.delete.attributes},${arn.action.sgsn.get.fd
n},${arn.action.sgsn.none.found},${arn.action.sgsn.one.found},${arn.nodes.sgsn.n
odeType}
${filter.node},"Execute a modeled action on a Managed Object - MGW",${arn.nodes.
mgw.node1},${arn.action.mgw.check.delete.command},${arn.action.mgw.fdn},${arn.ac
tion.mgw.create.action},${arn.action.mgw.create.attributes},${arn.action.mgw.del
ete.action},${arn.action.mgw.delete.attributes},${arn.action.mgw.get.fdn},${arn.
action.mgw.none.found},${arn.action.mgw.one.found},${arn.nodes.mgw.nodeType}
${filter.node},"Execute a modeled action on a Managed Object - RNC",${arn.nodes.
rnc.node1},${arn.action.rnc.check.delete.command},${arn.action.rnc.fdn},${arn.ac
tion.rnc.create.action},${arn.action.rnc.create.attributes},${arn.action.rnc.del
ete.action},${arn.action.rnc.delete.attributes},${arn.action.rnc.get.fdn},${arn.
action.rnc.none.found},${arn.action.rnc.one.found},${arn.nodes.rnc.nodeType}
```

To add a new node, add a new row to `gatActionArnLoop`, for example:

```
${filter.node},"Execute a modeled action on a Managed Object - MGW",${arn.nodes.
mgw.node1},${arn.action.mgw.check.delete.command},${arn.action.mgw.fdn},${arn.ac
tion.mgw.create.action},${arn.action.mgw.create.attributes},${arn.action.mgw.del
ete.action},${arn.action.mgw.delete.attributes},${arn.action.mgw.get.fdn},${arn.
action.mgw.none.found},${arn.action.mgw.one.found},${arn.nodes.mgw.nodeType}
```

Some of the values in `gatActionArnLoop` represent a property in `/root/agat/tdm.properties`, they are:

- `filter.node`
- `arn.nodes.<nodeType>.node<number>`
- `arn.action.<nodeType>.check.delete.command`
- `arn.action.<nodeType>.fdn`
- `arn.action.<nodeType>.create.action`
- `arn.action.<nodeType>.create.attributes`
- `arn.action.<nodeType>.delete.action`
- `arn.action.<nodeType>.delete.attributes`
- `arn.action.<nodeType>.get.fdn`
- `arn.action.<nodeType>.none.found`
- `arn.action.<nodeType>.one.found`
- `arn.nodes.<nodeType>.nodeType`

Configure these properties in `/root/agat/tdm.properties`. See `/root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties` template and Appendix B for examples.

Adding Test Data For: GAT-CM: Create and Delete Managed Objects on NE

The configuration is done in `/root/agat/tdm.properties` file and `gatCmeditCreateAndDeleteArnLoop` data source.

Check if the below properties are already configured in `/root/agat/tdm.properties`, otherwise add/edit it:

```
dataprovider.gatCmeditCreateAndDeleteArnLoop.type=tdm
dataprovider.gatCmeditCreateAndDeleteArnLoop.id=5c6fe43c36d1a00001253e50
```

The `dataprovider.gatCmeditCreateAndDeleteArnLoop.id` property must be pointing to `gatCmeditCreateAndDeleteArnLoop` data source ID.

Each node represents a new row in `gatCmeditCreateAndDeleteArnLoop`, as in the example:

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<MO_to_use>,<attribute_ →
to_use>,<new_value>,<another_new_value>,<FDN_of_MO_to_use>,<List_of_MO_attribute →
s_and_values>,nodeType
${filter.node},"Create and Delete Managed Objects on NE - ERBS",${arn.nodes.erbs →
.node1},${arn.get.set.erbs.mo},${arn.get.set.erbs.mo.attribute},${arn.get.set.er →
bs.mo.attribute.value},${arn.get.set.erbs.mo.attribute.new.value},${arn.get.set. →
erbs.create.fdn},${arn.get.set.erbs.create.fdn.attributes},${arn.nodes.erbs.nety →
pe}
${filter.node},"Create and Delete Managed Objects on NE - Radionode",${arn.nodes →
.radio.node1},${arn.get.set.radio.mo},${arn.get.set.radio.mo.attribute},${arn.ge →
t.set.radio.mo.attribute.value},${arn.get.set.radio.mo.attribute.new.value},${ar →
n.get.set.radio.create.fdn},${arn.get.set.radio.create.fdn.attributes},${arn.nod →
es.radio.netype}
${filter.node},"Create and Delete Managed Objects on NE - RBS",${arn.nodes.rbs.n →
ode1},${arn.get.set.rbs.mo},${arn.get.set.rbs.mo.attribute},${arn.get.set.rbs.mo →
.attribute.value},${arn.get.set.rbs.mo.attribute.new.value},${arn.get.set.rbs.cr →
eate.fdn},${arn.get.set.rbs.create.fdn.attributes},${arn.nodes.rbs.netype}
${filter.node},"Create and Delete Managed Objects on NE - RNC",${arn.nodes.rnc.n →
ode1},${arn.get.set.rnc.mo},${arn.get.set.rnc.mo.attribute},${arn.get.set.rnc.mo →
.attribute.value},${arn.get.set.rnc.mo.attribute.new.value},${arn.get.set.rnc.cr →
eate.fdn},${arn.get.set.rnc.create.fdn.attributes},${arn.nodes.rnc.netype}
${filter.node},"Create and Delete Managed Objects on NE - SGSN",${arn.nodes.sgsn →
.node1},${arn.get.set.sgsn.mo},${arn.get.set.sgsn.mo.attribute},${arn.get.set.sg →
sn.mo.attribute.value},${arn.get.set.sgsn.mo.attribute.new.value},${arn.get.set. →
sgsn.create.fdn},${arn.get.set.sgsn.create.fdn.attributes},${arn.nodes.sgsn.nety →
pe}
${filter.node},"Create and Delete Managed Objects on NE - MGW",${arn.nodes.mgw.n →
ode1},${arn.get.set.mgw.mo},${arn.get.set.mgw.mo.attribute},${arn.get.set.mgw.mo →
.attribute.value},${arn.get.set.mgw.mo.attribute.new.value},${arn.get.set.mgw.cr →
eate.fdn},${arn.get.set.mgw.create.fdn.attributes},${arn.nodes.mgw.netype}
```

To add a new node, add a new row to `gatCmeditCreateAndDeleteArnLoop`, for example:

```
${filter.node},"Create and Delete Managed Objects on NE - MGW",${arn.nodes.mgw.n →
ode1},${arn.get.set.mgw.mo},${arn.get.set.mgw.mo.attribute},${arn.get.set.mgw.mo →
.attribute.value},${arn.get.set.mgw.mo.attribute.new.value},${arn.get.set.mgw.cr →
eate.fdn},${arn.get.set.mgw.create.fdn.attributes},${arn.nodes.mgw.netype}
```

Some of the values in `gatCmeditCreateAndDeleteArnLoop` represent a property in `/root/agat/tdm.properties`, they are:

— `filter.node`

— `arn.nodes.<nodeType>.node<number>`

- arn.get.set.<nodeType>.mo
- arn.get.set.<nodeType>.mo.attribute
- arn.get.set.<nodeType>.mo.attribute.value
- arn.get.set.<nodeType>.mo.attribute.new.value
- arn.get.set.<nodeType>.create.fdn
- arn.get.set.<nodeType>.create.fdn.attributes
- arn.nodes.<nodeType>.netype

Configure these properties in /root/agat/tdm.properties. See /root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.propertiestemplate and Appendix B for examples.

Adding Test Data For: GAT-CM: Get and Set Attributes on a Managed Object on NE

The configuration is done in /root/agat/tdm.properties file and gatCmeditGetAndSetArnLoop data source.

Check if the following properties are already configured in /root/agat/tdm.properties. If not, add to or edit the file.

```
dataprovider.gatCmeditGetAndSetArnLoop.type=tdm
dataprovider.gatCmeditGetAndSetArnLoop.id=5c6fe45436d1a00001253e60
```

The dataprovider.gatCmeditGetAndSetArnLoop.id property must be pointing to gatCmeditGetAndSetArnLoop data source ID.

Each node represents a new row in gatCmeditGetAndSetArnLoop, as in the example:

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<MO_to_use>,<attribute_ →
to_use>,<new_value>,<another_new_value>,<FDN_of_MO_to_use>,<List_of_MO_attribute →
s_and_values>,nodeType
${filter.node},"Get and Set Attributes on a Managed Object on NE - ERBS",${arn.n →
odes.erbs.node1},${arn.get.set.erbs.mo},${arn.get.set.erbs.mo.attribute},${arn.g →
et.set.erbs.mo.attribute.value},${arn.get.set.erbs.mo.attribute.new.value},${arn →
.get.set.erbs.create.fdn},${arn.get.set.erbs.create.fdn.attributes},${arn.nodes. →
erbs.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE - Radionode",${ →
arn.nodes.radio.node1},${arn.get.set.radio.mo},${arn.get.set.radio.mo.attribute} →
,${arn.get.set.radio.mo.attribute.value},${arn.get.set.radio.mo.attribute.new.va →
lue},${arn.get.set.radio.create.fdn},${arn.get.set.radio.create.fdn.attributes}, →
${arn.nodes.radio.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE - RBS",${arn.no →
des.rbs.node1},${arn.get.set.rbs.mo},${arn.get.set.rbs.mo.attribute},${arn.get.s →
et.rbs.mo.attribute.value},${arn.get.set.rbs.mo.attribute.new.value},${arn.get.s →
et.rbs.create.fdn},${arn.get.set.rbs.create.fdn.attributes},${arn.nodes.rbs.nety →
pe}
${filter.node},"Get and Set Attributes on a Managed Object on NE - RNC",${arn.no →
des.rnc.node1},${arn.get.set.rnc.mo},${arn.get.set.rnc.mo.attribute},${arn.get.s →
et.rnc.mo.attribute.value},${arn.get.set.rnc.mo.attribute.new.value},${arn.get.s →
et.rnc.create.fdn},${arn.get.set.rnc.create.fdn.attributes},${arn.nodes.rnc.nety →
pe}
```

```
${filter.node},"Get and Set Attributes on a Managed Object on NE - SGSN",${arn.n →
odes.sgsn.node1},${arn.get.set.sgsn.mo},${arn.get.set.sgsn.mo.attribute},${arn.g →
et.set.sgsn.mo.attribute.value},${arn.get.set.sgsn.mo.attribute.new.value},${arn →
.get.set.sgsn.create.fdn},${arn.get.set.sgsn.create.fdn.attributes},${arn.nodes. →
sgsn.netype}
${filter.node},"Get and Set Attributes on a Managed Object on NE - MGW",${arn.no →
des.mgw.node1},${arn.get.set.mgw.mo},${arn.get.set.mgw.mo.attribute},${arn.get.s →
et.mgw.mo.attribute.value},${arn.get.set.mgw.mo.attribute.new.value},${arn.get.s →
et.mgw.create.fdn},${arn.get.set.mgw.create.fdn.attributes},${arn.nodes.mgw.nety →
pe}
```

To add a new node, add a new row to `gatCmeditGetAndSetArnLoop`, for example:

```
${filter.node},"Get and Set Attributes on a Managed Object on NE - RNC",${arn.no →
des.rnc.node1},${arn.get.set.rnc.mo},${arn.get.set.rnc.mo.attribute},${arn.get.s →
et.rnc.mo.attribute.value},${arn.get.set.rnc.mo.attribute.new.value},${arn.get.s →
et.rnc.create.fdn},${arn.get.set.rnc.create.fdn.attributes},${arn.nodes.rnc.nety →
pe}t.set.sgsn.create.fdn.attributes},${arn.nodes.sgsn.netype}
```

Some of the values in `gatCmeditGetAndSetArnLoop` represent a property in `/root/agat/tdm.properties`, they are:

- `filter.node`
- `arn.nodes.<nodeType>.node<number>`
- `arn.get.set.<nodeType>.mo`
- `arn.get.set.<nodeType>.mo.attribute`
- `arn.get.set.<nodeType>.mo.attribute.value`
- `arn.get.set.<nodeType>.mo.attribute.new.value`
- `arn.get.set.<nodeType>.create.fdn`
- `arn.get.set.<nodeType>.create.fdn.attributes`
- `arn.nodes.<nodeType>.netype`

Configure these properties in `/root/agat/tdm.properties`. See `/root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties` template and Appendix B for examples.

Adding Test Data For: Copy NE to a Non-Live Configuration, Modify NE Data, Activate Configuration and Undo Changes

The configuration is done in `/root/agat/tdm.properties` file and `gatCopyModifyActivateUndoConfigArnLoop` data source.

Check if the following properties are already configured in `/root/agat/tdm.properties`. If not, add to or edit the file.

```
dataprovider.gatCopyModifyActivateUndoConfigArnLoop.type=tdm
dataprovider.gatCopyModifyActivateUndoConfigArnLoop.id=5c6fe4a136d1a00001253e88
```

The `dataprovider.gatCopyModifyActivateUndoConfigArnLoop.id` property must be pointing to `gatCopyModifyActivateUndoConfigArnLoop` data source ID.

Each node represents a new row in `gatCopyModifyActivateUndoConfigArnLoop`, as in the example:

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<configName>,<MO_to_use →
>,<attribute_to_use>,<new_value>,<another_new_value>,nodeType
${filter.node},"Copy NE to a non live configuration, Modify NE data, Activate co →
nfiguration and Undo Changes - ERBS",${arn.nodes.erbs.node1},${arn.config.erbs.n →
ame},${arn.config.erbs.mo},${arn.config.erbs.mo.attribute},${arn.config.erbs.mo. →
attribute.value},${arn.config.erbs.mo.attribute.new.value},${arn.nodes.erbs.nety →
pe}
${filter.node},"Copy NE to a non live configuration, Modify NE data, Activate co →
nfiguration and Undo Changes - Radionode",${arn.nodes.radio.node1},${arn.config. →
radio.name},${arn.config.radio.mo},${arn.config.radio.mo.attribute},${arn.config →
.radio.mo.attribute.value},${arn.config.radio.mo.attribute.new.value},${arn.node →
s.radio.netype}
${filter.node},"Copy NE to a non live configuration, Modify NE data, Activate co →
nfiguration and Undo Changes - RBS",${arn.nodes.rbs.node1},${arn.config.rbs.name →
},${arn.config.rbs.mo},${arn.config.rbs.mo.attribute},${arn.config.rbs.mo.attrib →
ute.value},${arn.config.rbs.mo.attribute.new.value},${arn.nodes.rbs.netype}
${filter.node},"Copy NE to a non live configuration, Modify NE data, Activate co →
nfiguration and Undo Changes - RNC",${arn.nodes.rnc.node1},${arn.config.rnc.name →
},${arn.config.rnc.mo},${arn.config.rnc.mo.attribute},${arn.config.rnc.mo.attrib →
ute.value},${arn.config.rnc.mo.attribute.new.value},${arn.nodes.rnc.netype}
${filter.node},"Copy NE to a non live configuration, Modify NE data, Activate co →
nfiguration and Undo Changes - SGSN",${arn.nodes.sgsn.node1},${arn.config.sgsn.n →
ame},${arn.config.sgsn.mo},${arn.config.sgsn.mo.attribute},${arn.config.sgsn.mo. →
attribute.value},${arn.config.sgsn.mo.attribute.new.value},${arn.nodes.sgsn.nety →
pe}
${filter.node},"Copy NE to a non live configuration, Modify NE data, Activate co →
nfiguration and Undo Changes - MGW",${arn.nodes.mgw.node1},${arn.config.mgw.name →
},${arn.config.mgw.mo},${arn.config.mgw.mo.attribute},${arn.config.mgw.mo.attrib →
ute.value},${arn.config.mgw.mo.attribute.new.value},${arn.nodes.mgw.netype}
```

To add a new node, add a new row to `gatCopyModifyActivateUndoConfigArnLoop`, for example:

```
${filter.node},"Copy NE to a non live configuration, Modify NE data, Activate co →
nfiguration and Undo Changes - Radionode",${arn.nodes.radio.node1},${arn.config. →
radio.name},${arn.config.radio.mo},${arn.config.radio.mo.attribute},${arn.config →
.radio.mo.attribute.value},${arn.config.radio.mo.attribute.new.value},${arn.node →
s.radio.netype}
```

Some of the values in `gatCopyModifyActivateUndoConfigArnLoop` represent a property in `/root/agat/tdm.properties`, they are:

- `filter.node`
- `arn.nodes.<nodeType>.node<number>`
- `arn.config.<nodeType>.name`
- `arn.config.<nodeType>.mo`
- `arn.config.<nodeType>.mo.attribute`

— arn.config.<nodeType>.mo.attribute.value

— arn.config.<nodeType>.mo.attribute.new.value

— arn.nodes.<nodeType>.netype

Configure these properties in /root/agat/tdm.properties. See /root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties template and Appendix B for examples.

Adding Test Data For: GAT-CM: Create a Non-Live Configuration and List Configurations

The configuration is done in /root/agat/tdm.properties file and gatCreateListConfigArnLoop data source.

Check if the following properties are already configured in /root/agat/tdm.properties. If not, add to or edit the file.

```
dataprovider.gatCreateListConfigArnLoop.type=tdm
dataprovider.gatCreateListConfigArnLoop.id=5c6fe4b836d1a00001253e98
```

The dataprovider.gatCreateListConfigArnLoop.id property must be pointing to gatCreateListConfigArnLoop data source ID.

Each node represents a new row in gatCreateListConfigArnLoop, as in the example:

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<configName>,<MO_to_use →
>,<attribute_to_use>,<new_value>,<another_new_value>,nodeType
${filter.node},"Create a Non Live Configuration and List configurations - ERBS", →
${arn.nodes.erbs.node1},${arn.config.erbs.name},${arn.config.erbs.mo},${arn.conf →
ig.erbs.mo.attribute},${arn.config.erbs.mo.attribute.value},${arn.config.erbs.mo →
.attribute.new.value},${arn.nodes.erbs.netype}
${filter.node},"Create a Non Live Configuration and List configurations - Radion →
ode",${arn.nodes.radio.node1},${arn.config.radio.name},${arn.config.radio.mo},${ →
arn.config.radio.mo.attribute},${arn.config.radio.mo.attribute.value},${arn.conf →
ig.radio.mo.attribute.new.value},${arn.nodes.radio.netype}
${filter.node},"Create a Non Live Configuration and List configurations - RBS",$ →
{arn.nodes.rbs.node1},${arn.config.rbs.name},${arn.config.rbs.mo},${arn.config.r →
bs.mo.attribute},${arn.config.rbs.mo.attribute.value},${arn.config.rbs.mo.attrib →
ute.new.value},${arn.nodes.rbs.netype}
${filter.node},"Create a Non Live Configuration and List configurations - RNC",$ →
{arn.nodes.rnc.node1},${arn.config.rnc.name},${arn.config.rnc.mo},${arn.config.r →
nc.mo.attribute},${arn.config.rnc.mo.attribute.value},${arn.config.rnc.mo.attrib →
ute.new.value},${arn.nodes.rnc.netype}
${filter.node},"Create a Non Live Configuration and List configurations - SGSN", →
${arn.nodes.sgsn.node1},${arn.config.sgsn.name},${arn.config.sgsn.mo},${arn.conf →
ig.sgsn.mo.attribute},${arn.config.sgsn.mo.attribute.value},${arn.config.sgsn.mo →
.attribute.new.value},${arn.nodes.sgsn.netype}
${filter.node},"Create a Non Live Configuration and List configurations - MGW",$ →
{arn.nodes.mgw.node1},${arn.config.mgw.name},${arn.config.mgw.mo},${arn.config.m →
gw.mo.attribute},${arn.config.mgw.mo.attribute.value},${arn.config.mgw.mo.attrib →
ute.new.value},${arn.nodes.mgw.netype}
```

To add a new node, add a new row to gatCreateListConfigArnLoop, for example:

```
${filter.node},"Create a Non Live Configuration and List configurations - RNC",$ →
{arn.nodes.rnc.node1},${arn.config.rnc.name},${arn.config.rnc.mo},${arn.config.r →
```

```
nc.mo.attribute},${arn.config.rnc.mo.attribute.value},${arn.config.rnc.mo.attrib
ute.new.value},${arn.nodes.rnc.netype}
```

Some of the values in gatCreateListConfigArnLoop represent a property in /root/agat/tdm.properties, they are:

filter.node

arn.nodes.<nodeType>.node<number>

arn.config.<nodeType>.name

arn.config.<nodeType>.mo

arn.config.<nodeType>.mo.attribute

arn.config.<nodeType>.mo.attribute.value

arn.config.<nodeType>.mo.attribute.new.value

arn.nodes.<nodeType>.netype

Configure these properties in /root/agat/tdm.properties. See /root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties template and Appendix B for examples.

Adding Test Data For: GAT-CM: Delete a Non-Live Configuration

The configuration is done in /root/agat/tdm.properties file and gatDeleteConfigArnLoop data source.

Check if the following properties are already configured in /root/agat/tdm.properties. If not, add to or edit the file.

```
dataprovider.gatDeleteConfigArnLoop.type=tdm
dataprovider.gatDeleteConfigArnLoop.id=5c6fe4cd36d1a00001253ea8
```

The dataprovider.gatDeleteConfigArnLoop.id property must point to gatDeleteConfigArnLoop data source ID.

Each node represents a new row in gatDeleteConfigArnLoop, as in the example:

```
filterTestCaseByNodeNameOrNodeType,testCaseId,<nodeName>,<configName>,<MO_to_use
>,<attribute_to_use>,<new_value>,<another_new_value>,nodeType
${filter.node},"Delete a Non Live Configuration - ERBS",${arn.nodes.erbs.node1},
${arn.config.erbs.name},${arn.config.erbs.mo},${arn.config.erbs.mo.attribute},${
arn.config.erbs.mo.attribute.value},${arn.config.erbs.mo.attribute.new.value},${
arn.nodes.erbs.netype}
${filter.node},"Delete a Non Live Configuration - Radionode",${arn.nodes.radio.n
ode1},${arn.config.radio.name},${arn.config.radio.mo},${arn.config.radio.mo.attr
ibute},${arn.config.radio.mo.attribute.value},${arn.config.radio.mo.attribute.ne
w.value},${arn.nodes.radio.netype}
${filter.node},"Delete a Non Live Configuration - RBS",${arn.nodes.rbs.node1},${
arn.config.rbs.name},${arn.config.rbs.mo},${arn.config.rbs.mo.attribute},${arn.c
onfig.rbs.mo.attribute.value},${arn.config.rbs.mo.attribute.new.value},${arn.nod
```

```
es.rbs.netype}
${filter.node},"Delete a Non Live Configuration - RNC",${arn.nodes.rnc.node1},${ →
arn.config.rnc.name},${arn.config.rnc.mo},${arn.config.rnc.mo.attribute},${arn.c →
onfig.rnc.mo.attribute.value},${arn.config.rnc.mo.attribute.new.value},${arn.nod →
es.rnc.netype}
${filter.node},"Delete a Non Live Configuration - SGSN",${arn.nodes.sgsn.node1}, →
${arn.config.sgsn.name},${arn.config.sgsn.mo},${arn.config.sgsn.mo.attribute},${ →
arn.config.sgsn.mo.attribute.value},${arn.config.sgsn.mo.attribute.new.value},${ →
arn.nodes.sgsn.netype}
${filter.node},"Delete a Non Live Configuration - MGW",${arn.nodes.mgw.node1},${ →
arn.config.mgw.name},${arn.config.mgw.mo},${arn.config.mgw.mo.attribute},${arn.c →
onfig.mgw.mo.attribute.value},${arn.config.mgw.mo.attribute.new.value},${arn.nod →
es.mgw.netype}
```

To add a new node, add a new row to `gatDeleteConfigArnLoop`, for example:

```
${filter.node},"Delete a Non Live Configuration - RNC",${arn.nodes.rnc.node1},${ →
arn.config.rnc.name},${arn.config.rnc.mo},${arn.config.rnc.mo.attribute},${arn.c →
onfig.rnc.mo.attribute.value},${arn.config.rnc.mo.attribute.new.value},${arn.nod →
es.rnc.netype}
```

Some of the values in `gatDeleteConfigArnLoop` represent a property in `/root/agat/tdm.properties`, they are:

- `filter.node`
- `arn.nodes.<nodeType>.node<number>`
- `arn.config.<nodeType>.name`
- `arn.config.<nodeType>.mo`
- `arn.config.<nodeType>.mo.attribute`
- `arn.config.<nodeType>.mo.attribute.value`
- `arn.config.<nodeType>.mo.attribute.new.value`
- `arn.nodes.<nodeType>.netype`

Configure these properties in `/root/agat/tdm.properties`. See `/root/.m2/repository/0_TestDataTemplateFiles/properties/tdm.properties` template and Appendix B for examples.

9 Appendix B - CM Properties on tdm.properties Example

The `CM_DeleteNode` suite has a parameter that indicates whether to delete the whole subnetwork or not.

This parameter must be added for every node used in the suite, as follows:

```
<network element name>.deleteSubNetwork=<value>
```

The options for *<value>* are `true` or `false`.

Example

This appendix shows an example of CM properties that must be configured on `tdm.properties`.

```
################################################################################
#######################        CM SPECIFIC DATA         #######################
################################################################################
# CM: General TDM datasources
dataprovider.gatHeartbeatAndManualSyncArnLoop.type=tdm
dataprovider.gatHeartbeatAndManualSyncArnLoop.id=5c6fe4e836d1a00001253eb8
dataprovider.gatCmeditGetAndSetArnLoop.type=tdm
dataprovider.gatCmeditGetAndSetArnLoop.id=5c6fe45436d1a00001253e60
dataprovider.gatCmeditCreateAndDeleteArnLoop.type=tdm
dataprovider.gatCmeditCreateAndDeleteArnLoop.id=5c6fe43c36d1a00001253e50
dataprovider.gatCreateListConfigArnLoop.type=tdm
dataprovider.gatCreateListConfigArnLoop.id=5c6fe4b836d1a00001253e98
dataprovider.gatCopyModifyActivateUndoConfigArnLoop.type=tdm
dataprovider.gatCopyModifyActivateUndoConfigArnLoop.id=5c6fe4a136d1a00001253e88
dataprovider.gatDeleteConfigArnLoop.type=tdm
dataprovider.gatDeleteConfigArnLoop.id=5c6fe4cd36d1a00001253ea8
dataprovider.generateArnlImportFileDataSource.type=tdm
dataprovider.generateArnlImportFileDataSource.id=5c6fe53436d1a00001253ede
dataprovider.gatActionArnLoop.type=tdm
dataprovider.gatActionArnLoop.id=5c4af2f536d1a000012539a3
dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.type=tdm
dataprovider.gatCmeditGetAndSetCreateDeleteArnLoop.id=5c6fe47536d1a00001253e70
################################################################################
####################      CM: TDM datasources by Node Type       ###############  →
#####
################################################################################
dataprovider.generateArnlErbsImportFile.type=tdm
dataprovider.generateArnlErbsImportFile.id=5c6fe51536d1a00001253ed8
dataprovider.generateArnlRadioImportFile.type=tdm
dataprovider.generateArnlRadioImportFile.id=5c6fe56036d1a00001253ef4
dataprovider.generateArnlRbsImportFile.type=tdm
dataprovider.generateArnlRbsImportFile.id=5c6fe57a36d1a00001253efc
dataprovider.generateArnlMgwImportFile.type=tdm
dataprovider.generateArnlMgwImportFile.id=5c6fe54b36d1a00001253eee
dataprovider.generateArnlRNCImportFile.type=tdm
dataprovider.generateArnlRNCImportFile.id=5c6fe58e36d1a00001253f02
dataprovider.generateArnlSGSNImportFile.type=tdm
dataprovider.generateArnlSGSNImportFile.id=5c6fe5a136d1a00001253f08
# Create a new for each new Node Type.
#dataprovider.generateArnl<NodeType>ImportFile.type=tdm
#dataprovider.generateArnl<NodeType>ImportFile.id=<generateArnl<NodeType>ImportF  →
ile id>

################################################################################
####################        CM: General properties         ####################
################################################################################
```

```
arn.import.config.nonlive=importConfig
arn.import.config.live=Live
arn.import.dynamic.filetype=dynamic
arn.import.3gpp.filetype=3GPP
modifier.update=update
################################################################################
#######################      CM: NODE SPECIFIC DATA     #######################
################################################################################
# Node Data - RadioNode 1
arn.nodes.radio.node1=<network element name>
arn.nodes.radio.node1.ossprefix=SubNetwork=G2RBS\\,ManagedElement=${arn.nodes.ra
dio.node1}
#arn.nodes.radio.node2=<network element name>
#arn.nodes.radio.node2.ossprefix=<ossprefix>
# Node Data - ERBS 1
arn.nodes.erbs.node1=<network element name>
arn.nodes.erbs.node1.ossprefix=<ossprefix>
# Node Data - RBS 1
arn.nodes.rbs.node1=<network element name>
arn.nodes.rbs.node1.ossprefix=<ossprefix>
# Node Data - RNC 1
arn.nodes.rnc.node1=<network element name>
arn.nodes.rnc.node1.ossprefix=<ossprefix>
# Node Data - SGSN 1
arn.nodes.sgsn.node1=<network element name>
arn.nodes.sgsn.node1.ossprefix=<ossprefix>
# Node Data - MGW 1
arn.nodes.mgw.node1=<network element name>
arn.nodes.mgw.node1.ossprefix=<ossprefix>
# Node Data - <NodeType> 1
#arn.nodes.<NodeType>.node1=<network element name>
#arn.nodes.<NodeType>.node1.ossprefix=<ossprefix>
################################################################################
#######################       CM: NODE TYPE DATA        #######################
################################################################################
# CM: Node Type filter
filter.node=RadioNode;ERBS;RBS;RNC;SGSN;MGW
#######################       RadioNode NODE TYPE       #######################
arn.nodes.radio.netype=RadioNode
arn.get.set.radio.mo=ManagedElement
arn.get.set.radio.mo.attribute=userLabel
arn.get.set.radio.mo.attribute.value=some_new_value
arn.get.set.radio.mo.attribute.new.value=another_new_value
arn.get.set.radio.create.fdn=${arn.nodes.radio.node1.ossprefix}\\,SystemFunction
s=1\\,SecM=1\\,CertM=1\\,NodeCredential=3
arn.get.set.radio.create.fdn.attributes=nodeCredentialId=3
arn.config.radio.name=radioNodeChapter7Config
arn.config.radio.mo=ENodeBFunction
arn.config.radio.mo.attribute=userLabel
arn.config.radio.mo.attribute.value=a_new_value
arn.config.radio.mo.attribute.new.value=another_new_value
arn.import.radio.3gpp.filename=modify_ENodeBFunction_RadioNode_ArnLoop.xml
arn.import.radio.3gpp.filename.id=${arn.import.radio.3gpp.filename}
arn.import.radio.3gpp.fdn=${arn.nodes.radio.node1.ossprefix}\\,ENodeBFunction=1
arn.import.radio.mo.attribute=ENodeBFunction.userLabel
arn.import.radio.mo.to.change=userLabel
arn.import.radio.mo.attribute.value=EnodeBFunction userlabel import update
arn.action.radio.fdn=${arn.nodes.radio.node1} BrmBackupManager
arn.action.radio.check.delete.command=cmedit get ${arn.nodes.radio.node1} BrmBac
kup.BackupName==RadioNodeUniqueBackupName
arn.action.radio.create.action=createBackup
arn.action.radio.attributes=(name=RadioNodeUniqueBackupName)
arn.action.radio.delete.action=deleteBackup
arn.action.radio.get.fdn=${arn.nodes.radio.node1} BrmBackup.BackupName==RadioNod
eUniqueBackupName
arn.action.radio.none.found=0 inst
arn.action.radio.one.found=1 inst
arn.verify.non.persisted.set.get.radio.mo=EUtranCellFDD
arn.verify.non.persisted.set.get.radio.mo.attribute=<mo attribute value>
arn.verify.non.persisted.set.get.radio.fdn=${arn.nodes.radio.node1.ossprefix}\\,
ENodeBFunction=1\\,EUtranCellFDD=1

#######################         ERBS NODE TYPE          #######################
arn.nodes.erbs.netype=ERBS
arn.get.set.erbs.mo=ManagedElement
arn.get.set.erbs.mo.attribute=userLabel
arn.get.set.erbs.mo.attribute.value=some_new_value
arn.get.set.erbs.mo.attribute.new.value=another_new_value
arn.get.set.erbs.create.fdn=${arn.nodes.erbs.node1.ossprefix}\\,ManagedElement=1
\\,SwManagement=1\\,LoadModule=testLoadModule
```

```
arn.get.set.erbs.create.fdn.attributes=<some attributes>
arn.config.erbs.name=erbsChapter7Config
arn.config.erbs.mo=ENodeBFunction
arn.config.erbs.mo.attribute=userLabel
arn.config.erbs.mo.attribute.value=a_new_value
arn.config.erbs.mo.attribute.new.value=another_new_value
arn.import.erbs.dynamic.filename=modify_ENodeBFunction_ERBS_ArnLoop.csv
arn.import.erbs.dynamic.filename.id=${arn.import.erbs.dynamic.filename}
arn.import.erbs.dynamic.fdn=${arn.nodes.erbs.node1.ossprefix}\\,ManagedElement=1 →
\\,ENodeBFunction=1
arn.import.erbs.mo.attribute=ENodeBFunction.userLabel
arn.import.erbs.mo.to.change=userLabel
arn.import.erbs.mo.attribute.value=EnodeBFunction userlabel import update
arn.action.erbs.fdn=${arn.nodes.erbs.node1} ConfigurationVersion
arn.action.erbs.check.delete.command=cmedit action ${arn.nodes.erbs.node1} Confi →
gurationVersion delete.(configurationVersionName=ERBSUniqueBackupName)
arn.action.erbs.create.action=create
arn.action.erbs.create.attributes=(configurationVersionName=ERBSUniqueBackupName →
\\,identity=Test01\\,type=STANDARD\\,operatorName=test\\,comment=TestComment01)
arn.action.erbs.delete.action=delete
arn.action.erbs.delete.attributes=(configurationVersionName=ERBSUniqueBackupName →
)
arn.action.erbs.get.fdn=${arn.nodes.erbs.node1} ConfigurationVersion.*
arn.action.erbs.none.found=CVDoesNotExistsException
arn.action.erbs.one.found=UniqueBackup

arn.verify.non.persisted.set.get.erbs.mo=EUtranCellFDD
arn.verify.non.persisted.set.get.erbs.mo.attribute=<mo attribute value>
arn.verify.non.persisted.set.get.erbs.fdn=${arn.nodes.erbs.node1.ossprefix}\\,Ma →
nagedElement=1\\,ENodeBFunction=1\\,EUtranCellFDD=1

########################          RBS NODE TYPE          ########################
arn.nodes.rbs.netype=RBS
arn.get.set.rbs.mo=ManagedElement
arn.get.set.rbs.mo.attribute=userLabel
arn.get.set.rbs.mo.attribute.value=some_new_value
arn.get.set.rbs.mo.attribute.new.value=another_new_value
arn.get.set.rbs.create.fdn=${arn.nodes.rbs.node1.ossprefix}\\,ManagedElement=1\\ →
,SwManagement=1\\,LoadModule=testLoadModule
arn.get.set.rbs.create.fdn.attributes=LoadModuleId=testLoadModule\\,loadModuleFi →
lePath=path\\,productData=(productRevision=rev\\,productName=name\\,productInfo= →
info\\,productNumber=123\\,productionDate=20150325)
arn.config.rbs.name=rbsChapter7Config
arn.config.rbs.mo=NodeBFunction
arn.config.rbs.mo.attribute=userLabel
arn.config.rbs.mo.attribute.value=a_new_value
arn.config.rbs.mo.attribute.new.value=another_new_value
arn.import.rbs.dynamic.filename=modify_NodeBFunction_RBS_ArnLoop.csv
arn.import.rbs.dynamic.filename.id=${arn.import.rbs.dynamic.filename}
arn.import.rbs.dynamic.fdn=${arn.nodes.rbs.node1.ossprefix}\\,ManagedElement=1\\ →
,NodeBFunction=1
arn.import.rbs.mo.attribute=NodeBFunction.userLabel
arn.import.rbs.mo.to.change=userLabel
arn.import.rbs.mo.attribute.value=NodeBFunction userlabel import update
arn.action.rbs.fdn=${arn.nodes.rbs.node1} ConfigurationVersion
arn.action.rbs.check.delete.command=cmedit action ${arn.nodes.rbs.node1} Configu →
rationVersion delete.(configurationVersionName=RBSUniqueBackupName)
arn.action.rbs.create.action=create
arn.action.rbs.create.attributes=(configurationVersionName=RBSUniqueBackupName\\ →
,identity=Test01\\,type=STANDARD\\,operatorName=test\\,comment=TestComment01)
arn.action.rbs.delete.action=delete
arn.action.rbs.delete.attributes=(configurationVersionName=RBSUniqueBackupName)
arn.action.rbs.get.fdn=${arn.nodes.rbs.node1} ConfigurationVersion.*
arn.action.rbs.none.found=CVDoesNotExistsException
arn.action.rbs.one.found=UniqueBackup
arn.verify.non.persisted.set.get.rbs.mo=NodeBFunction
arn.verify.non.persisted.set.get.rbs.mo.attribute=dlLicFractBbPool2
arn.verify.non.persisted.set.get.rbs.fdn=${arn.nodes.rbs.node1.ossprefix}\\,Mana →
gedElement=1\\,NodeBFunction=1

########################          RNC NODE TYPE          ########################
arn.nodes.rnc.netype=RNC
arn.nodes.rnc.nodeType=RNCNode
arn.get.set.rnc.mo=ManagedElement
arn.get.set.rnc.mo.attribute=userLabel
arn.get.set.rnc.mo.attribute.value=some_new_value
arn.get.set.rnc.mo.attribute.new.value=another_new_value
arn.get.set.rnc.create.fdn=${arn.nodes.rnc.node1.ossprefix}\\,ManagedElement=1\\ →
,SwManagement=1\\,LoadModule=testLoadModule
arn.get.set.rnc.create.fdn.attributes=LoadModuleId=testLoadModule\\,loadModuleFi →
```

```
lePath=path\\,productData=(productRevision=rev\\,productName=name\\,productInfo=
info\\,productNumber=123\\,productionDate=20150325)
arn.config.rnc.name=rncChapter7Config
arn.config.rnc.mo=RncFunction
arn.config.rnc.mo.attribute=userLabel
arn.config.rnc.mo.attribute.value=a_new_value
arn.config.rnc.mo.attribute.new.value=another_new_value
arn.import.rnc.dynamic.filename=modify_RncFunction_RNCNode_ArnLoop.csv
arn.import.rnc.dynamic.filename.id=${arn.import.rnc.dynamic.filename}
arn.import.rnc.dynamic.fdn=${arn.nodes.rnc.node1.ossprefix}\\,ManagedElement=1\\
,RncFunction=1
arn.import.rnc.mo.attribute=RncFunction.userLabel
arn.import.rnc.mo.to.change=userLabel
arn.import.rnc.mo.attribute.value=RncFunction userlabel import update
arn.action.rnc.fdn=${arn.nodes.rnc.node1} ConfigurationVersion
arn.action.rnc.check.delete.command=cmedit action ${arn.nodes.rnc.node1} Configu
rationVersion delete.(configurationVersionName=RNC24UniqueBackupName)
arn.action.rnc.create.action=create
arn.action.rnc.create.attributes=(configurationVersionName=RNCUniqueBackupName\\
,identity=Test01\\,type=STANDARD\\,operatorName=test\\,comment=TestComment01)
arn.action.rnc.delete.action=delete
arn.action.rnc.delete.attributes=(configurationVersionName=RNCUniqueBackupName)
arn.action.rnc.get.fdn=${arn.nodes.rnc.node1} ConfigurationVersion.*
arn.action.rnc.none.found=CVDoesNotExistsException
arn.action.rnc.one.found=UniqueBackup
arn.verify.non.persisted.set.get.rnc.mo=RncFunction
arn.verify.non.persisted.set.get.rnc.mo.attribute=nwInitCallReestTime
arn.verify.non.persisted.set.get.rnc.fdn=${arn.nodes.rnc.node1.ossprefix}\\,Mana
gedElement=1\\,RncFunction=1

######################          SGSN NODE TYPE          ######################
arn.nodes.sgsn.netype=SGSN
arn.get.set.sgsn.mo=ManagedElement
arn.get.set.sgsn.mo.attribute.value=some_new_value
arn.get.set.sgsn.mo.attribute.new.value=another_new_value
arn.get.set.sgsn.create.fdn=${arn.nodes.sgsn.node1.ossprefix}\\,ManagedElement=$
{arn.nodes.sgsn.node1}\\,SwManagement=1\\,LoadModule=testLoadModule
arn.get.set.sgsn.create.fdn.attributes=LoadModuleId=testLoadModule\\,loadModuleF
ilePath=path\\,productData=(productRevision=rev\\,productName=name\\,productInfo
=info\\,productNumber=123\\,productionDate=20150325)
arn.config.sgsn.name=sgsnChapter7Config
arn.config.sgsn.mo=SgsnFunction
arn.config.sgsn.mo.attribute=userLabel
arn.config.sgsn.mo.attribute.value=a_new_value
arn.config.sgsn.mo.attribute.new.value=another_new_value
arn.import.sgsn.dynamic.filename=modify_SgsnFunction_SGSNNode_ArnLoop.csv
arn.import.sgsn.dynamic.filename.id=${arn.import.sgsn.dynamic.filename}
arn.import.sgsn.dynamic.fdn=${arn.nodes.sgsn.node1.ossprefix}\\,${arn.config.sgs
n.mo}=1
arn.import.sgsn.mo.attribute=${arn.config.sgsn.mo}.userLabel
arn.import.sgsn.mo.to.change=userLabel
arn.import.sgsn.mo.attribute.value=${arn.config.sgsn.mo} userlabel import update
arn.action.sgsn.fdn=${arn.nodes.sgsn.node1} ConfigurationVersion
arn.action.sgsn.check.delete.command=cmedit action ${arn.nodes.sgsn.node1} Confi
gurationVersion delete.(configurationVersionName=SGSN24UniqueBackupName)
arn.action.sgsn.create.action=create
arn.action.sgsn.create.attributes=(configurationVersionName=SGSNUniqueBackupName
\\,identity=Test01\\,type=STANDARD\\,operatorName=test\\,comment=TestComment01)
arn.action.sgsn.delete.action=delete
arn.action.sgsn.delete.attributes=(configurationVersionName=SGSNUniqueBackupName
)
arn.action.sgsn.get.fdn=${arn.nodes.sgsn.node1} ConfigurationVersion.*
arn.action.sgsn.none.found=CVDoesNotExistsException
arn.action.sgsn.one.found=UniqueBackup
arn.verify.non.persisted.set.get.sgsn.mo=SgsnFunction
arn.verify.non.persisted.set.get.sgsn.mo.attribute=nwInitCallReestTime
arn.verify.non.persisted.set.get.sgsn.fdn=${arn.nodes.sgsn.node1.ossprefix}\\,Sg
snFunction=1

######################          MGW NODE TYPE          ######################
arn.nodes.mgw.netype=MGW
arn.nodes.mgw.nodeType=MGW
arn.get.set.mgw.mo=ManagedElement
arn.get.set.mgw.mo.attribute=userLabel
arn.get.set.mgw.mo.attribute.value=some_new_value
arn.get.set.mgw.mo.attribute.new.value=another_new_value
arn.get.set.mgw.create.fdn=${arn.nodes.mgw.node1.ossprefix}\\,ManagedElement=1\\
,SwManagement=1\\,LoadModule=testLoadModule
arn.get.set.mgw.create.fdn.attributes=LoadModuleId=testLoadModule\\,loadModuleFi
lePath=path\\,productData=(productRevision=rev\\,productName=name\\,productInfo=
```

```
info\\,productNumber=123\\,productionDate=20150325)
arn.config.mgw.name=mgwChapter7Config
arn.config.mgw.mo=MgwApplication
arn.config.mgw.mo.attribute=userLabel
arn.config.mgw.mo.attribute.value=a_new_value
arn.config.mgw.mo.attribute.new.value=another_new_value
arn.import.mgw.dynamic.filename=modify_${arn.config.mgw.mo}_${arn.nodes.mgw.netype}Node_ArnLoop.csv
arn.import.mgw.dynamic.filename.id=${arn.import.mgw.dynamic.filename}
arn.import.mgw.dynamic.fdn=${arn.nodes.mgw.node1.ossprefix}\\,ManagedElement=${arn.nodes.mgw.node1}\\,${arn.config.mgw.mo}=1
arn.import.mgw.mo.attribute=${arn.config.mgw.mo}.userLabel
arn.import.mgw.mo.to.change=userLabel
arn.import.mgw.mo.attribute.value=${arn.config.mgw.mo} userlabel import update
arn.action.mgw.fdn=${arn.nodes.mgw.node1} ConfigurationVersion
arn.action.mgw.check.delete.command=cmedit action ${arn.nodes.mgw.node1} ConfigurationVersion delete.(configurationVersionName=MGW24UniqueBackupName)
arn.action.mgw.create.action=create
arn.action.mgw.create.attributes=(configurationVersionName=MGWUniqueBackupName\\,identity=Test01\\,type=STANDARD\\,operatorName=test\\,comment=TestComment01)
arn.action.mgw.delete.action=delete
arn.action.mgw.delete.attributes=(configurationVersionName=MGWUniqueBackupName)
arn.action.mgw.get.fdn=${arn.nodes.mgw.node1} ConfigurationVersion.*
arn.action.mgw.none.found=CVDoesNotExistsException
arn.action.mgw.one.found=UniqueBackup
arn.verify.non.persisted.set.get.mgw.mo=MgwApplication
arn.verify.non.persisted.set.get.mgw.mo.attribute=rtcpBearerSupervisionTimerVoip
arn.verify.non.persisted.set.get.mgw.fdn=${arn.nodes.mgw.node1.ossprefix}\\,${arn.verify.non.persisted.set.get.mgw.mo}=1

#######################         <New> NODE TYPE         ######################
#arn.nodes.<nodeType>.netype=<nodeType>
#
#arn.get.set.<nodeType>.mo=
#arn.get.set.<nodeType>.mo.attribute.value=
#arn.get.set.<nodeType>.mo.attribute.new.value=
#arn.get.set.<nodeType>.create.fdn=
#arn.get.set.<nodeType>.create.fdn.attributes=
#
#arn.config.<nodeType>.name=
#arn.config.<nodeType>.mo=
#arn.config.<nodeType>.mo.attribute=
#arn.config.<nodeType>.mo.attribute.value=
#arn.config.<nodeType>.mo.attribute.new.value=
#
#arn.import.<nodeType>.dynamic.filename=
#arn.import.<nodeType>.dynamic.filename.id=
#arn.import.<nodeType>.dynamic.fdn=
#arn.import.<nodeType>.mo.attribute=
#arn.import.<nodeType>.mo.to.change=
#arn.import.<nodeType>.mo.attribute.value=
#
#arn.action.<nodeType>.fdn=
#arn.action.<nodeType>.check.delete.command=
#arn.action.<nodeType>.create.action=
#arn.action.<nodeType>.create.attributes=
#arn.action.<nodeType>.delete.action=
#arn.action.<nodeType>.delete.attributes=
#arn.action.<nodeType>.get.fdn=
#arn.action.<nodeType>.none.found=
#arn.action.<nodeType>.one.found=
#
#arn.verify.non.persisted.set.get.<nodeType>.mo=
#arn.verify.non.persisted.set.get.<nodeType>.mo.attribute=
#arn.verify.non.persisted.set.get.<nodeType>.fdn=
```

10 Appendix C - Customize AGAT Test Suite

> This appendix describes how to select the individual test cases you wish to run from a specific test suite. This is done by modifying both the `agat_schedule.xml` and the individual `suite.xml` files.

Modify the Schedule File

1. Navigate to `/root/agat` and open the `agat_schedule_xxxxxxxx.xml` file.
2. Select the test suite for which specific test cases are to be run in the schedule file.
3. Change the *<suites>* attribute with the path of the suite XML file. For example, Agat_NHM test suite in schedule file can be changed as shown:

Default:

```
<item timeout-in-seconds="1800" stop-on-fail="false">
    <name>Agat_NHM</name>
    <component>com.ericsson.oss.services.nhm:ERICTAFkpicalculationservice_CXP903 →
1364:1.61.6</component>
    <suites>Agat_NHM.xml</suites>
    <groups>ENM_EXTERNAL_TESTWARE</groups>
</item>
```

Modified:

```
<item timeout-in-seconds="1800" stop-on-fail="false">
    <name>Agat_NHM</name>
    <component>com.ericsson.oss.services.nhm:ERICTAFkpicalculationservice_CXP903 →
1364:1.61.6</component>
    <suites>/root/testdata/suites/Agat_NHM.xml</suites>
    <groups>ENM_EXTERNAL_TESTWARE</groups>
</item>
```

Modify the Suite XML File

After changing the schedule XML file for a specific test suite, the suite XML file needs to be changed to either include or exclude individual test cases that belong to the test suite.

1. Navigate to `/root/testdata/suites` and open the suite XML file.
2. Use the 'include' or 'exclude' keywords to choose which test cases to run from this suite. The following example shows the "deleteKpiForTests" test case excluded from the Agat_NHM test suite.

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Agat_NHM">
    <parameter name="dataprovidername" value="nodesToAdd"/>
    <parameter name="datasource" value="nodesToAdd"/>
```

```
    <test name="NHM Setup Teardown">
        <classes>
            <class name="com.ericsson.oss.services.nhm.cases.setup.SetupAndTeard
ownScenario"/>
        </classes>
    </test>
    <test name="NHM KPI Test">
        <classes>
            <class name="com.ericsson.oss.services.nhm.cases.scenarios.NhmRealNo
deScenario">
                <methods>
                    <include name="createKpiForTests"/>
                    <include name="verifyBreachTest"/>
                    <exclude name="deleteKpiForTests"/>
                </methods>
            </class>
        </classes>
    </test>
</suite>
```

Executing the Tests

Note: Because some test cases depend on others, excluding a particular test case can cause others to fail.

For example, in PM Statistical testware, if the *Creation of a Statistical Subscription* test case is excluded, then the *Activation of a Statistical Subscription* test case fails. For this reason, take particular care in relation to which test cases are being excluded.

The execution of the test is the same as shown in Execute Tests on page 28.

11 Appendix D - Manual Cleanup for AGAT Test Data

When a test case has timed out or broken, and when test cases do not run cleanup tasks, test data created during test execution can remain on the ENM system.

This data can include users created by testware, PM subscriptions created by PM testware, and so on.

To perform a manual cleanup after a testware failure, follow the instructions provided:

General Cleanup

This section describes manual cleanup that is necessary for any AGAT testware.

To clean up the ENM users created by AGAT testware, complete the following steps:

1. Log on to the ENM GUI.
2. Go to **User Management**.
3. Check the list of users for any that have a username prefix that appears in the **username** column in the `UsersToCreate` data source.
4. Delete these users.

Figure 15 Users Created by AGAT Test and agatAdm Prefix Value in the Username Column in UsersToCreate

Testware-Specific Cleanup

This section describes manual cleanup that is necessary for the related testware.

PMIC Testwares:

Problem PMIC subscriptions can be left over in ENM.

Suites Affected
- Agat_PM_Statistical
- Agat_PM_Celltrace

To clean up, proceed as follows:

1. Log on to the ENM GUI.
2. Go to the **PM Initiation and Collection** page.
3. Search for subscriptions with "`AGAT_<random number>`" in the name.
4. If the subscription is active, deactivate it.
5. Remove the subscription.

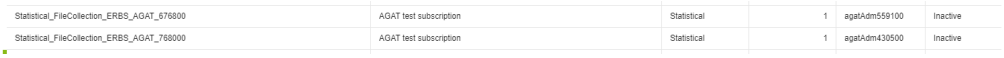


Figure 16 Example: Statistical_FileCollection_ERBS_AGAT_676800 Was Created by AGAT and Should Be Removed.

12 Appendix E - Migrating from remoteInfo to TDM

1. Log on to the TDM and edit the `NodeData.sample` and `UsersToCreate.sample` data sources with required node and user details.
2. Data sources can be edited by importing the existing CSV files using the import option. To do this, click **Edit**.
3. Save the edited file.

Note: For more information, click **Help** on the TAF **Test Data Management** screen.

13 Troubleshooting

This section describes the possible problems which may be encountered during the tests and gives hints about how to handle these situations.

13.1 Tests Fail with "RuntimeException: org.supercsv.exception.SuperCsvException"

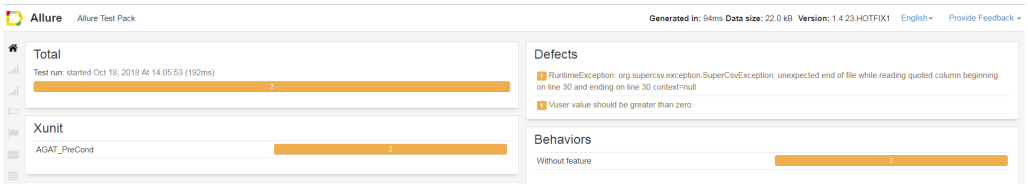


Figure 17 Runtime Exception

Solution

1. Check the NodeData data source for any missing column or character.

13.2 Tests Fail with "Vuser value should be greater than zero" Error

> Test fails with "Vuser value should be greater than zero" and shows the below error in the Allure report.

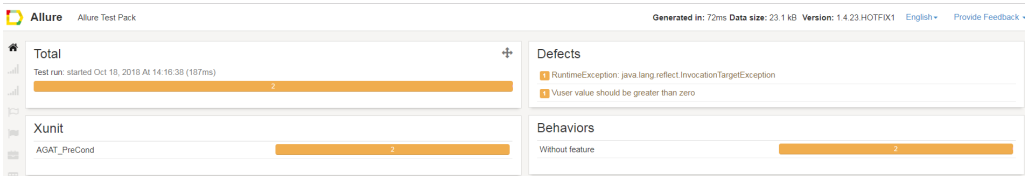


Figure 18 Vuser Value Should be Greater Than Zero

Solution

1. Check the `UsersToCreate` column for missing characters.

13.3 Tests Fail with "IllegalArgumentException: Failed to find an applicable data source for name 'availableUsers'" Error

> Tests fail with "IllegalArgumentException: Failed to find an applicable data source for name 'availableUsers'" and shows the below error in the Allure report.

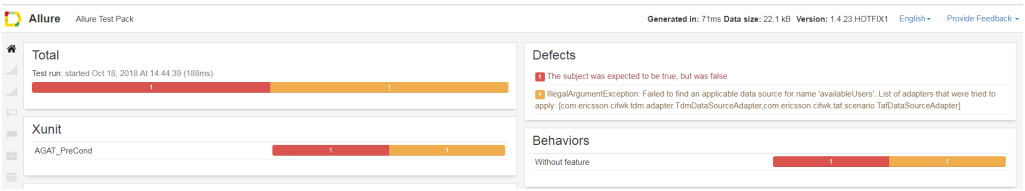


Figure 19 Illegal Argument Exception

Solution

1. If the tests fail with this type of error, possible causes to address are:
   - The ENM HAProxy is not reachable from the AGAT server.
   - The user specified in the `tdm.properties` file with the property `ci-user` does not have `ADMINISTRATOR` and `SECURITY_ADMIN` role.
   - The password for `ci-user` is wrong.
   - The IP address or hostname of HAProxy is not correct in the `host.properties.json` file.

13.4 Test Failure Caused by Testware Dependency Issue

> The test fails because of a testware dependency not present on taf_te_slave container. The console logs show the following error:

```
[INFO] Scanning for projects...
[INFO] Downloading: https://arm101-eiffel004.lmera.ericsson.se:8443/nexus/conten →
t/groups/public/com/ericsson/oss/shm/ERICTAFshm_CXP9030703/1.58.6-AFR-17-10-SNAP →
SHOT/maven-metadata.xml
[WARNING] Failed to create parent directories for tracking file /root/.m2/reposi →
tory/com/ericsson/oss/shm/ERICTAFshm_CXP9030703/1.58.6-AFR-17-10-SNAPSHOT/resolv →
er-status.properties
[WARNING] Could not transfer metadata com.ericsson.oss.shm:ERICTAFshm_CXP9030703 →
:1.58.6-AFR-17-10-SNAPSHOT/maven-metadata.xml from/to nexus (https://arm101-eiff →
el004.lmera.ericsson.se:8443/nexus/content/groups/public): /root/.m2/repository/ →
com/ericsson/oss/shm/ERICTAFshm_CXP9030703/1.58.6-AFR-17-10-SNAPSHOT/maven-metad →
```

```
ata-nexus.xml.part.lock (No such file or directory)
[INFO] Downloading: https://arm101-eiffel004.lmera.ericsson.se:8443/nexus/conten →
t/groups/public/com/ericsson/oss/shm/ERICTAFshm_CXP9030703/1.58.6-AFR-17-10-SNAP →
SHOT/ERICTAFshm_CXP9030703-1.58.6-AFR-17-10-SNAPSHOT.pom
[WARNING] Failed to create parent directories for tracking file /root/.m2/reposi →
tory/com/ericsson/oss/shm/ERICTAFshm_CXP9030703/1.58.6-AFR-17-10-SNAPSHOT/ERICTA →
Fshm_CXP9030703-1.58.6-AFR-17-10-SNAPSHOT.pom.lastUpdated
[ERROR] [ERROR] Some problems were encountered while processing the POMs:
[ERROR] Non-resolvable import POM: Could not transfer artifact com.ericsson.oss. →
shm:ERICTAFshm_CXP9030703:pom:1.58.6-AFR-17-10-SNAPSHOT from/to nexus (https://a →
rm101-eiffel004.lmera.ericsson.se:8443/nexus/content/groups/public): /root/.m2/r →
epository/com/ericsson/oss/shm/ERICTAFshm_CXP9030703/1.58.6-AFR-17-10-SNAPSHOT/E →
RICTAFshm_CXP9030703-1.58.6-AFR-17-10-SNAPSHOT.pom.part.lock (No such file or di →
rectory) @ line 31, column 25
 @
[ERROR] The build could not read 1 project -> [Help 1]
```

Solution

1. Verify that the `taf_te_slave` container is restarted after mounting the AGAT ISO.
2. Verify that the dependencies are present in the `/root/.m2/repository/`. To help in identifying these dependencies, refer to the errors on the `taf_te_slave` container in Allure.
3. Navigate into the container with the following command:

```
[root@localhost~]# docker exec -it `docker ps -q -f 'NAME=taf_te_slave'` /bi →
n/bash
                    root@4c95c0c1d6e9:~#
```

4. Navigate to the `/root/.m2/repository/` directory on the container.

```
root@4c95c0c1d6e9:~# cd /root/.m2/repository/
```

5. Verify that the file structure and versions in this location on the container are the same on the localhost directory `/root/.m2/repository/`.

```
                    root@4c95c0c1d6e9:~# ls /root/.m2/repository/
                    0_TestDataTemplateFiles           backport-util-concurre →
nt  commons-chain           commons-httpclient  de          javax       log4j   →
   oro       xerces
                    agat_schedule_20180621181941.xml  c3p0                    →
    commons-cli             commons-io          dom4j       jaxen       logkit  →
   rr_moved  xml-apis
                    antlr                             cglib                   →
    commons-codec           commons-lang        eu          jgraph      mysql   →
   ru
                    aopalliance                       classworlds             →
    commons-collections     commons-logging     iaik        jline       nekohtm →
l  sshtools
                    asm                               com                     →
    commons-configuration   commons-pool        io          joda-time   net     →
   sslext
                    avalon-framework                  commons-beanutils       →
    commons-digester        commons-validator   javassist   junit       org     →
   xalan
```

6. Once confirmed, exit from the container.

```
root@4c95c0c1d6e9:~# exit
```

7. Copy the `suitesExtraction.sh` script from `/root/.m2/repository/0_TestDataTemplateFiles/` to `/root/agat/`.

```
cp /root/.m2/repository/0_TestDataTemplateFiles/suitesExtraction.sh /root/agat
```

8. Give executable permission to the `suitesExtraction.sh` script.

```
chmod 777 suitesExtraction.sh
```

9. Run the `suitesExtraction.sh` script.

```
./suitesExtraction.sh
```

Option 1

— copy the file `agat_schedule_<timestamp>.xml` with the name `agat_schedule.xml` and execute script

```
./suitesExtraction.sh
```

This script creates the folder `/root/testdata/suites`, to which all suites XML files are copied.

Option 2

— Execute script with `agat_schedule_<timestamp>.xml` as parameter:

```
./suitesExtraction.sh agat_schedule_<timestamp>.xml
```

This script creates the folder `/root/testdata/suites`, to which all suites XML files are copied.

Note: The two options are equivalent: without parameters the name of the default file (agat_schedule.xml) is taken, otherwise the xml file, passed onto as parameter, is used.

10. Log on to the TDM UI application (`https://<TE address>:9443/`) with the user credentials added while configuring your LDAP user during the installation of TE.

11. Verify that all the data source files are properly restored.

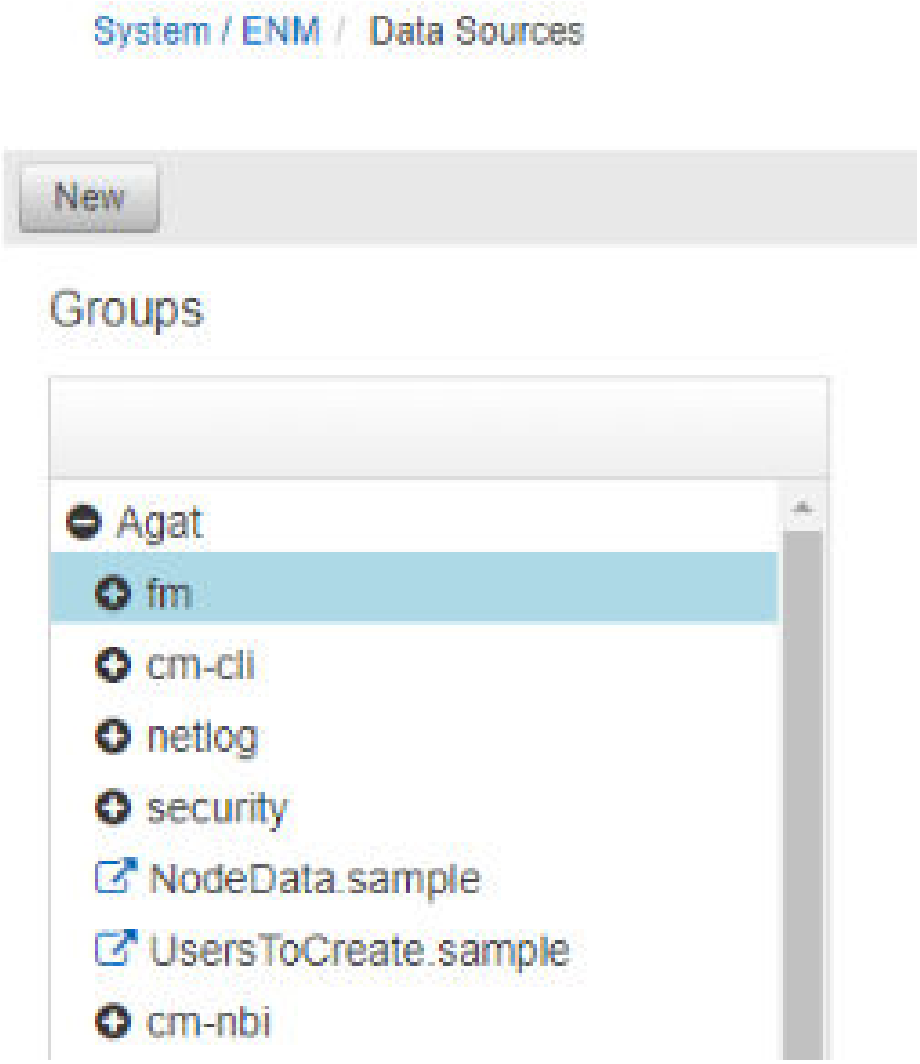


Figure 20 Restored Data Source Files

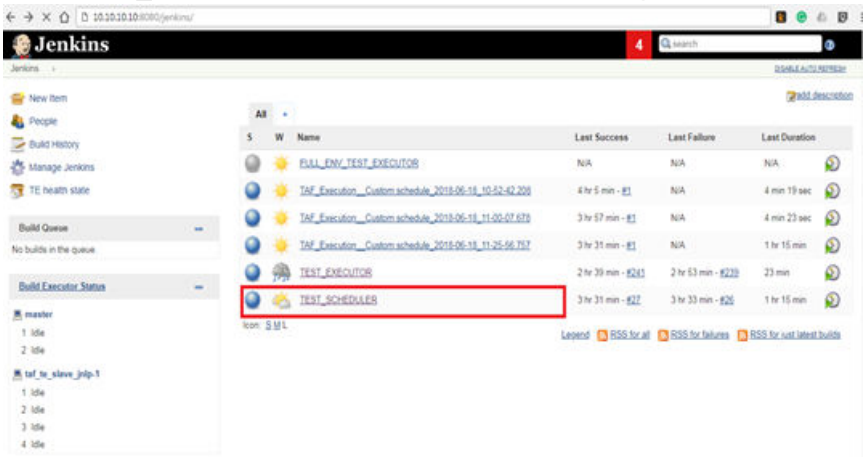
13.5 Get Allure Report in Case of Loss of Terminal

If for any reason the terminal is closed or terminated, the Allure report can be accessed by two methods:

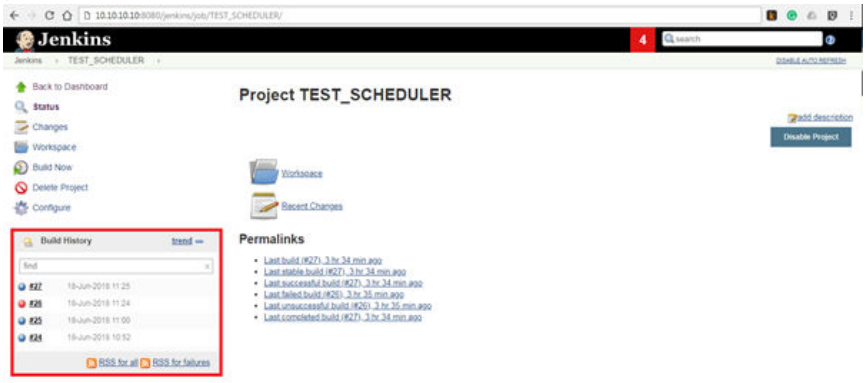
Steps

1. Option 1
    a. Navigate to Jenkins web URL: `http://<TE address>:8080/jenkins/`

b. Click **TEST_SCHEDULER** as shown in the screen capture:



c. In the **Build History**, identify and click the build which is triggered as part of the AGAT test execution:



d. Click **Console Output** and find the `ALLURE_LOG_URL` keyword in the Console Output logs:

e. Copy the URL and open it in the browser to get the Allure report.

2. Option 2

a. Log on to TE.

b. Navigate to the TE logs directory:

```
[root@atvts3186 ~]# cd /var/log/te_logs/
```

c. Search for all logs sorted by date, most recent last:

```
[root@atvts3186 te_logs]# ls -ltr
total 124
drwxrwxrwx. 14 root root 4096 Jun  1 16:25 f5799339-a3c7-4337-92b4-
9dbfe2a0dd17
drwxrwxrwx.  3 root root 4096 Jun  1 17:15 fe348954-ea6f-447f-9a57-
d05ef8641f00
drwxrwxrwx. 14 root root 4096 Jun  1 17:59 5331dd6e-fc6e-4dc3-b9ce-
d6de6314408e
```

d. Find the logs directory with the time stamp of test execution and copy the directory name. This is specific to the particular test execution.

e. Access the Allure report in the browser with the link:

```
http://<IP Address>:8088/<Logs Directory Name>/
Example - http://<IP Address>:8088/5331dd6e-fc6e-4dc3-b9ce-d6de6314
408e/
```

Reference List

[1] *Test Executor Configuration and Operation*, 1531-CNA 403 4011

Available from local Ericsson support.

[2] *ENM Security System Administrator Guide*, 2/1543-AOM 901 151

[3] *Release Independence GAT*, 6/197 03-9/FCP 130 2854 Uen

Available from local Ericsson support.

[4] *ENM Network Security Configuration System Administrator Guide*, 2/1543-AOM 901 151-2