Oracle® Communications

Cloud Native Core Automated Testing Suite Guide

Release 25.2.202
G52929-01
March 2026

Oracle Communications Cloud Native Core Automated Testing Suite Guide, Release 25.2.202

G52929-01

Contents

Preface

Documentation Accessibility i
Diversity and Inclusion i
Conventions i

1 Introduction

1.1 Overview 1
1.2 Deployment Model 2
1.3 References 3

2 ATS Framework Features

2.1 ATS API 2
2.1.1 Generating an API Token for a User 2
2.1.2 Use the RESTful Interfaces 4
2.1.2.1 Configuring Host 5
2.1.2.2 Starting Jobs 6
2.1.2.3 Monitoring Jobs 15
2.1.2.4 Stopping Jobs 16
2.1.2.5 Getting Test Suite Artifacts 18
2.2 ATS Custom Abort 21
2.3 ATS Feature Activation and Deactivation 25
2.4 ATS GUI Enhancements 29
2.5 ATS Health Check 31
2.6 ATS Jenkins Job Queue 33
2.7 Application Log Collection 34
2.7.1 Application Log Collection and Parallel Test Execution Integration 36
2.8 ATS Maintenance Scripts 37
2.8.1 ATS Scripts 38
2.8.2 Updating ats_install.sh 39
2.8.3 Restarting Jenkins without Restarting Pod 39
2.8.4 Updating Stub Scripts 40
2.8.5 Running ATS and Stub Deployment Scripts 41

2.9 ATS System Name and Version Display on the ATS GUI 41
2.10 ATS Tagging Support 41
2.10.1 Combination of Tags and their Results 43
2.11 Custom Folder Implementation 46
2.12 Health Check 47
2.13 Individual Stage Group Selection 56
2.14 Lightweight Performance 57
2.14.1 Configure <NF>-Performance Pipeline 57
2.15 Managing Final Summary Report, Build Color, and Application Log 58
2.16 Modifying Login Password 66
2.17 Multiselection Capability for Features and Scenarios 66
2.17.1 Feature Level Selection 66
2.17.2 Scenario Selection 67
2.18 Parallel Test Execution 68
2.18.1 ATS GUI Page Changes 71
2.18.2 ATS Console Log Changes 71
2.18.3 Downloading or Viewing Individual Group Logs 72
2.19 Parameterization 75
2.19.1 Running Test Cases 76
2.20 PCAP Log Collection 78
2.20.1 Application Log Collection and Parallel Test Execution Integration 78
2.21 Persistent Volume for 5G ATS 79
2.21.1 Processing Flow 79
2.21.2 Deploying Persistent Volume 80
2.21.3 Backward Porting 84
2.22 Single Click Job Creation 84
2.22.1 Configuring Single Click Job 84
2.23 Support for ATS Deployment in OCI 87
2.23.1 Accessing ATS GUI in OCI 87
2.23.2 Adding Ingress and Egress Rules to Access the OCI Console 88
2.24 Support for Transport Layer Security 92
2.24.1 Deploy ATS with TLS Enabled 93
2.24.1.1 Generate JKS File for Jenkins Server 93
2.24.1.2 Enable ATS GUI with HTTPS 98
2.25 Test Results Analyzer 100
2.25.1 Accessing Test Results Analyzer Feature 100
2.26 Support for Test Case Mapping and Count 103
2.26.1 Access Test Case Mapping and Count Feature 103

3 Installing ATS for Different Network Functions

3.1 Installing ATS for BSF 1

3.1.1 Resource Requirements 1
3.1.2 Downloading the ATS Package 4
3.1.3 Deploy ATS with TLS Enabled 5
3.1.3.1 Generate JKS File for Jenkins Server 5
3.1.3.2 Enable TLS on Python Stubs 9
3.1.3.3 Enable TLS on Diam Sim 11
3.1.3.4 Enable ATS GUI with HTTPS 13
3.1.4 Deploy BSF ATS with Authenticated Prometheus communication 15
3.1.5 Pushing the Images to Customer Docker Registry 15
3.1.6 Configuring ATS 17
3.1.6.1 Enabling Static Port 17
3.1.6.2 Enable Static API Node Port 17
3.1.6.3 Service Account Requirements 18
3.1.6.4 Enabling Aspen Service Mesh 18
3.1.6.5 Enabling Health Check 21
3.1.6.6 Enabling Persistent Volume 23
3.1.6.7 ATS-BSF API Extended Support 25
3.1.6.8 Configuration for Dual Stack Support 26
3.1.7 Deploying ATS and Pods 26
3.1.7.1 Deploying ATS in Kubernetes Cluster 26
3.1.7.2 Deploying ocstub-py Stubs in Kubernetes Cluster 28
3.1.7.3 Deploying DNS Stub in Kubernetes Cluster 33
3.1.7.4 Deploying ocdiam Simulator in Kubernetes Cluster 35
3.1.8 Post-Installation Steps 36
3.2 Installing ATS for NSSF 48
3.2.1 Resource Requirements 48
3.2.2 Locating and Downloading ATS and Simulator Images 50
3.2.3 Deploying ATS in Kubernetes Cluster 52

4 Running NF Test Cases Using ATS

4.1 Running BSF Test Cases using ATS 1
4.1.1 Prerequisites 1
4.1.2 Logging into ATS 5
4.1.3 Running BSF_NewFeatures Pipeline 7
4.1.4 BSF_NewFeatures Documentation 12
4.1.5 Running BSF Regression Pipeline 14
4.1.6 BSF_Regression Documentation 15
4.1.7 Running BSF_HealthCheck Pipeline 15
4.2 Running NSSF Test Cases using ATS 16
4.2.1 Prerequisites 16
4.2.2 Logging into ATS 17

4.2.3 NSSF-NewFeatures Pipeline 19
4.2.4 NSSF-NewFeatures Documentation 28
4.2.5 NSSF-Regression Pipeline 30
4.2.6 NSSF-Regression Documentation 40

Preface

- Documentation Accessibility
- Diversity and Inclusion
- Conventions

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

My Oracle Support

My Oracle Support (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support can assist you with My Oracle Support registration.

Call the Customer Access Support main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. When calling, make the selections in the sequence shown below on the Support telephone menu:

- For Technical issues such as creating a new Service Request (SR), select **1**.
- For Non-technical issues such as registration or assistance with My Oracle Support, select **2**.
- For Hardware, Networking and Solaris Operating System Support, select **3**.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

Acronyms

The following table lists the acronyms and the terminologies used in the document:

Table 1 Acronyms

Acronym	Meaning
ATS	Oracle Communications Cloud Native Core, Automated Test Suite
BDD	Behavior Driven Development
BSF	Oracle Communications Cloud Native Core, Binding Support Function
CI	Continuous Integration
CSR	Certificate Signing Request
EIR	Equipment Identity Register
Policy	Oracle Communications Cloud Native Core, Converged Policy
NRF	Oracle Communications Cloud Native Core, Network Repository Function
NSSF	Oracle Communications Cloud Native Core, Network Slice Selection Function
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
OL	Oracle Linux is a Linux distribution packaged and freely distributed by Oracle, available partially under the GNU (General Public License) since late 2006. It is compiled from Red Hat Enterprise Linux source code, replacing Red Hat branding with Oracle's.
PCF	Oracle Communications Cloud Native Core, Policy Control Function
PVC	Persistent Volume Claim
cnPCRF	Oracle Communications Cloud Native Core, Policy and Charging Rules Function
SAN	Subject Alternative Name
SCP	Oracle Communications Cloud Native Core, Service Communication Proxy
SEPP	Oracle Communications Cloud Native Core, Security Edge Protection Proxy
SLF	Subscriber Location Function
UDR	Oracle Communications Cloud Native Core, Unified Data Repository

What's New in This Guide

This section introduces the documentation updates for release 25.2.2xx.

ATS Framework Release 25.2.202 - G52929-01, March 2026

General Updates

- Added the following Helm parameters in Table 2-5 to manage Prometheus authenticated tokens:
  - `PrometheusAuthEnabled`
  - `PrometheusTLSEnabled`
- Added the `deploymentMode` Helm parameter, which supports the dual stack mechanism in ATS, in the following sections:
  - ATS API
  - Table 2-5
- Updated the ATS framework versions and their corresponding supporting NF versions in the Table 1-1.

BSF Release 25.2.200

- ATS release tag for BSF is changed to 25.2.200.
- Updated the following sections with 25.2.200 release updates under **Installing ATS for BSF**:
  - Downloading the ATS Package
  - Pushing the Images to Customer Docker Registry
- Added the Configuration for Dual Stack Support section to describe how to configure dual stack for ocats-bsf.
- Added the Configuration for dual stack support for ocstub-py in the Deploying ocstub-py Stubs in Kubernetes Cluster section.
- Added procedures to enable IPv6 and dual stack for ATS in the Post-Installation Steps section.
- Added the Deploy BSF ATS with Authenticated Prometheus communication section to describe the configuration required for authenticated Prometheus communication.
- Added a Note for the "p" and "q" parameters in the Running BSF_NewFeatures Pipeline section for authenticated Prometheus communication.
- Added the Enable TLS on Diam Sim section to enable TLS support for diam-sim.
- Updated the Managing Final Summary Report, Build Color, and Application Log section with the information about displaying the count and names of skipped features whenever skipped during a rerun.
- BSF ATS now supports tagging execution in merged execution, that is, support for Filter_With_Tags and Include_Regression / Include_NewFeatures. For more information, see Parallel Test Execution.
- BSF ATS now supports tag based run along with the stages and groups using ATS API. For more information, see ATS API.

- Number of new Test Cases added to pipeline: Provides a total of **2 feature files** and **8 scenarios** of BSF_NewFeatures pipeline.
- Previous Release Test Cases: Provides a total of **33 feature files** and **125 scenarios** of BSF_Regression pipeline.

NSSF Release 25.2.200

- ATS release tag for NSSF is changed to 25.2.200.
- Number of test cases in NSSF_NewFeatures pipeline: 45 new feature files with 784 new scenarios.
- Number of test cases in the NSSF_Regression pipeline: 7 feature files with 99 scenarios.
- Updated steps for Deploying ATS in Kubernetes Cluster.

1 Introduction

This document provides information about Automated Testing Suite (ATS) deployment model for 5G Network Functions (NFs).

ATS allows you to run software test cases using an automated testing tool. It compares the actual results with the expected or predicted results.

1.1 Overview

Using ATS, you can deploy and test 5G NFs.

This document provides information to implement ATS for the following 5G NFs:

- Oracle Communications Cloud Native Core, Binding Support Function (BSF)
- Oracle Communications Cloud Native Core, Network Repository Function (NRF)
- Oracle Communications Cloud Native Core, Network Slice Selection Function (NSSF)
- Oracle Communications Cloud Native Core, Converged Policy (Policy)
- Oracle Communications Cloud Native Core, Service Communication Proxy (SCP)
- Oracle Communications Cloud Native Core, Security Edge Protection Proxy (SEPP)
- Oracle Communications Cloud Native Core, Unified Data Repository (UDR)

ATS for 5G Network Functions

For 5G NFs, ATS is developed using Oracle Linux 8-slim as the base image. Jenkins is a part of the ATS image, and it provides a graphical user interface (GUI) to test either a single NF or multiple NFs independently in the same network environment.

ATS comprises NF docker images, ATS image, simulator images, and test cases of the specific NF. All these images and test cases constitute a fully automated suite to deploy and test NFs. You can combine it with any other Continuous Integration (CI) pipeline with minimal changes because 5G ATS uses Jenkins as GUI.

The ATS package contains the following elements:

- Test scripts and docker images of test container. The docker images have complete framework and libraries installed, which are common for all NFs working with the Behavior Driven Development (BDD) framework.
- Docker image of HTTP Server simulator.
- Helm chart to deploy the ATS (delivered as a tar file).
- Readme text file (.txt file).

ATS provides basic environment, framework, and a GUI (Jenkins) to run all the functional test cases.

ATS Framework Version with Supporting NF Version

Table 1-1 ATS Framework Version with Supporting NF

ATS Framework Version	BSF	NSSF
25.2.200	-	25.2.200
25.2.201	-	-
25.2.202	25.2.200	-

1.2 Deployment Model

According to the In-Cluster deployment model, ATS can coexist in the same cluster where the NFs are deployed. This deployment model is useful for In-Cluster testing.

Figure 1-1 In-Cluster Deployment Model in OCCNE

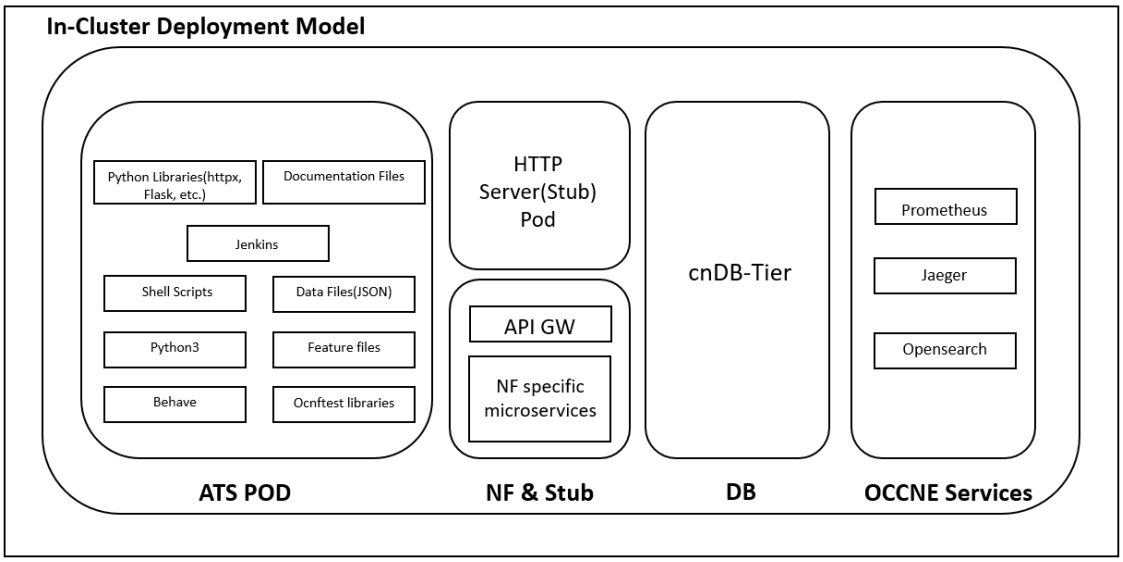
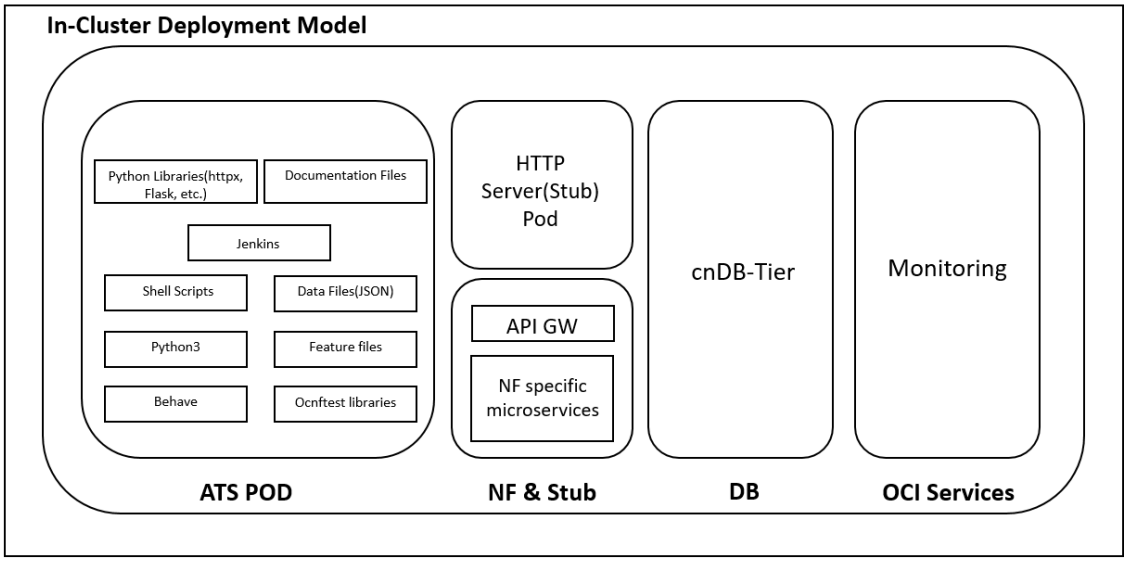


Figure 1-2 In-Cluster Deployment Model in OCI



1.3 References

For more information about NFs and their deployment processes, refer to the following documents:

- *Oracle Communications Cloud Native Core, Binding Support Function Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Configuration Console Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Network Repository Function Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Provisioning Gateway Guide*
- *Oracle Communications Cloud Native Core, Service Communication Proxy Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Policy Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Security Edge Protection Proxy Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Unified Data Repository Installation, Upgrade, and Fault Recovery Guide*
- *Oracle Communications Cloud Native Core, Cloud Native Environment Installation, Upgrade, and Fault Recovery Guide*

2 ATS Framework Features

This chapter describes ATS Framework features.

The following table lists ATS framework features supported by different NFs:

Table 2-1 ATS Framework Features Compliance Matrix

Features	BSF	NSSF
ATS API	Yes	No
ATS Custom Abort	Yes	Yes
ATS Feature Activation and Deactivation	Yes	No
ATS GUI Enhancements	Yes	Yes
ATS Health Check	Yes	No
ATS Jenkins Job Queue	Yes	No
Application Log Collection	Yes	No
ATS Maintenance Scripts	Yes	Yes
ATS System Name and Version Display on Jenkins GUI	Yes	Yes
ATS Tagging Support	No	No
Custom Folder Implementation	Yes	Yes
Health Check	Yes	Yes
Individual Stage Group Selection	Yes	No
Lightweight Performance	No	No
Managing Final Summary Report, Build Color, and Application Log	Yes	Yes
Modifying Login Password	Yes	Yes
Multiselection Capability for Features and Scenarios	Yes	No
Parallel Test Execution	Partially compliant (Parallel Test Execution Framework integrated, but only supports sequential execution)	Partially compliant (Parallel Test Execution Framework integrated, but only supports sequential execution)
Parameterization	Yes	No
PCAP Log Collection	No	No
Persistent Volume	Yes	Yes
Single Click Job Creation	Yes	Yes
Support for ATS Deployment in OCI	No	No
Support for Transport Layer Security	No	No
Test Result Analyzer	Yes	Yes
Test Case mapping and Count	Yes	No

2.1 ATS API

The Application Programming Interface (API) feature provides APIs to perform routine ATS tasks as follows:

- Start: To initiate one of the three test suites, such as Regression, New Features, or Performance.
- Monitor: To obtain the progress of a test suite's execution.
- Stop: To cancel an active test suite.
- Get Artifacts: To retrieve the JUNIT format XML test result files for a completed test suite.

> **Note**
>
> You can configure the `deploymentMode` Helm parameter, which supports the dual stack mechanism, to establish connections with ATS GUI and APIs. For more information, see Table 2-5.

For more information about configuring the tasks, see Use the RESTful Interfaces.

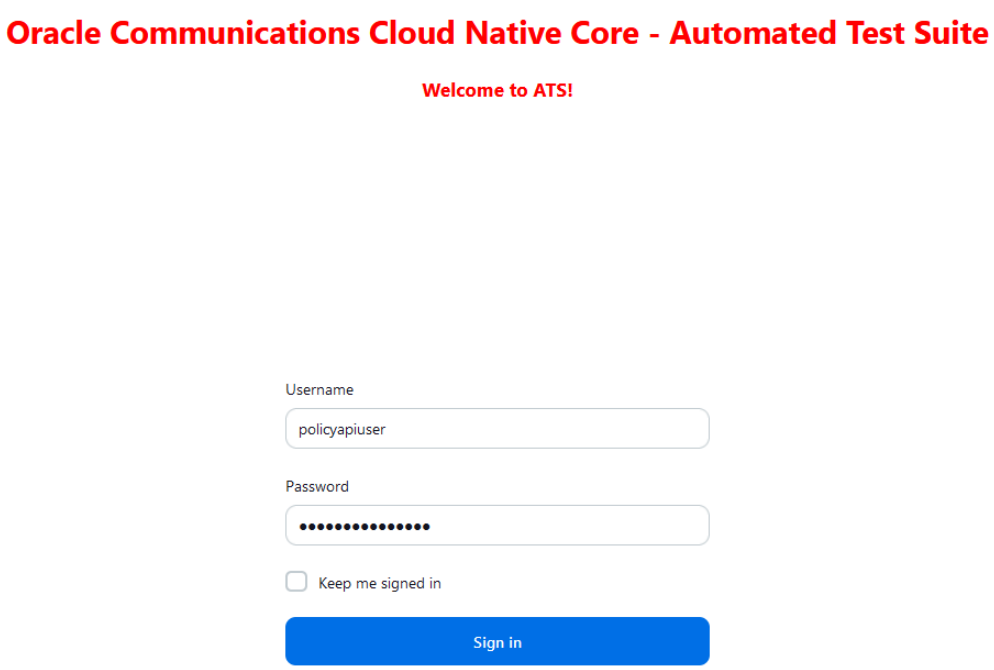
2.1.1 Generating an API Token for a User

An API token that has to be generated for the user to perform routine ATS tasks using the Restful Interfaces API. Any API call requires the use of an API token for authentication. You can generate the API token, and it works until it is revoked or deleted.

Perform the following procedure to generate an API token for a user:

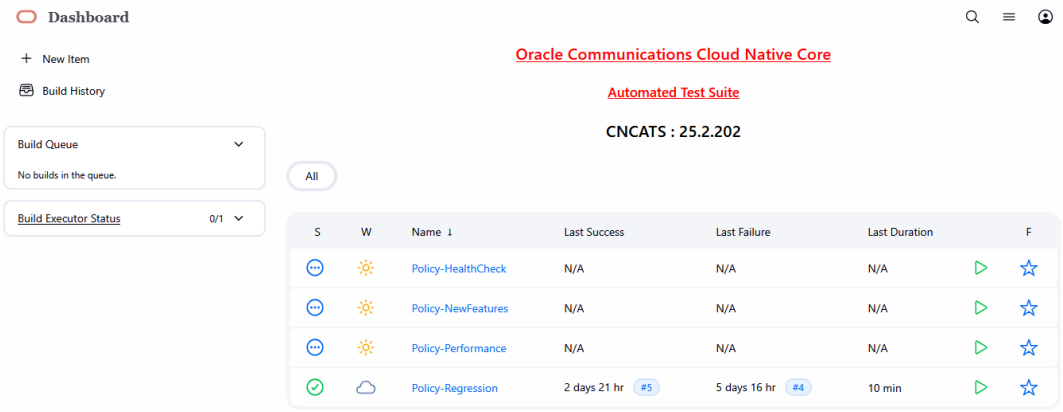
1. Log in to Jenkins as an NF API user to generate an API token.

Figure 2-1 ATS Login Page



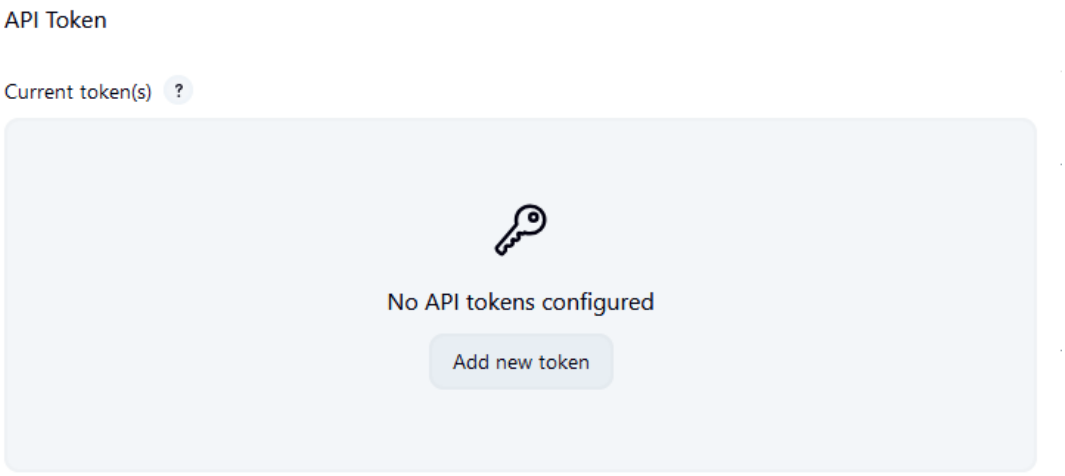
2. Click the user name in the upper right corner of the GUI, and then click **Security**.

Figure 2-2 Add Token



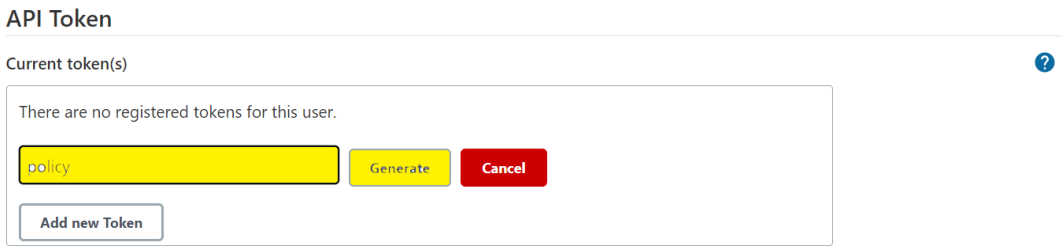
3. In the **API Token** section, click **Add new token**.

Figure 2-3 Add New Token



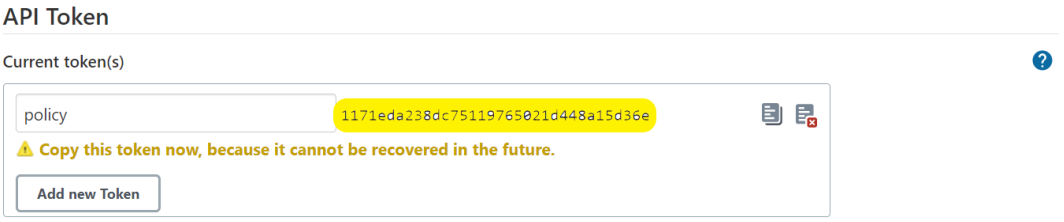
4. Enter a suitable name for the token, such as policy, and then click **Generate**.

Figure 2-4 Generate Token



5. Copy and save the generated token.
You cannot retrieve the token after closing the prompt.

Figure 2-5 Save Generated Token



6. Click **Save**.
An API token is generated and can be used for starting, monitoring, and stopping a job using the REST API.

2.1.2 Use the RESTful Interfaces

This section provides an overview of each RESTful interface.

2.1.2.1 Configuring Host

The host to access the ATS GUI will remain the same in non-OCI setup.

For OCI Setup

The following two ways are supported to access the ATS API in OCI:

Using Loadbalancer IP

1. Add proper Ingress/Egress security rules for ATS API port (5001) and ATS service nodeport corresponding to ATS API port in loadbalancer (nf_lb_subnet) and node subnet (nf_node_subnet). To add ingress and egress security rules, see the Adding Ingress and Egress Rules to Access the OCI Console section.
2. If this step is not performed to access GUI, then insert the following annotations under the **Metadata** section to assign an external IP (Loadbalancer IP):

   ```
   oci-network-load-balancer.oraclecloud.com/security-list-management-mode:
   None
   oci.oraclecloud.com/load-balancer-type: nlb
   ```

3. Edit and save the ATS service after ATS deployment:
   For example,

   ```
   kubectl edit svc ats-service-name -n
         ats-namespace
   ```

4. Access the GUI using URL: `<http/https>://<Loadbalancer IP>:5001`

> **Note**
>
> The assignment of Loadbalancer IP to the ATS service is subject to availability. If the Loadbalancer IP is not assigned to the ATS service even after applying the required annotations, try to debug on the OCI side.

Using Tunneling

1. Add an ingress security rule for the node subnet (nf_node_subnet) to allow TCP traffic on all ports from the operator subnet. To add ingress and egress security rules, see the Adding Ingress and Egress Rules to Access the OCI Console section.
2. Run the following ssh tunneling command from a bash terminal on your local PC:

   ```
   ssh -f -N -i <operator instance private key> -o StrictHostKeyChecking=no -
   o ProxyCommand="ssh -i <bastion private key> -o StrictHostKeyChecking=no -
   W %h:%p <bastion username>@<bastion IP>" <operator instance
   username>@<operator instance IP> -L <desired system port>:<Worker Node
   IP>:<ATS API NodePort> -o ServerAliveInterval=60 -o ServerAliveCountMax=300
   ```

   For example,`ssh -f -N -i id_rsa -o StrictHostKeyChecking=no -o ProxyCommand="ssh -i id_rsa -o StrictHostKeyChecking=no -W %h:%p opc@129.287.66.123" opc@10.1.76.7 -L 5009:10.9.60.118:32018 -o ServerAliveInterval=60 -o ServerAliveCountMax=300`

   Here, ATS GUI URLis http://localhost:5009

TROUBLESHOOTING

If the ATS API returns an error stating "Network is unreachable," ensure that there is a proper ingress security rule in the loadbalancer subnet (nf_lb_subnet) allowing traffic from the system where the ATS API is being utilized.

2.1.2.2 Starting Jobs

To start a job, use the following RESTful interfaces:

- Default Jenkins API: The default Jenkins API to start a pipeline job
- Custom API: To start a job forcibly

Starting a Job using Default Jenkins API
If any other job is already running, the job started with this API goes to Jenkins' job queue.

- Run the following command to start a job (Default Jenkins method):

```
curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/
buildWithParameters –user
      <username>:<API_token> --verbose
```

When the `atsGuiTLSEnabled` parameter is set to true:

```
curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/
buildWithParameters --user <username>:<API_token> --verbose --cacert
<path_to_root_certificate>
```

For example,

```
curl --request POST http://10.123.154.163:30427/job/Policy-NewFeatures/
buildWithParameters
      --user policyapiuser:111ad02d7471cec9ca689696e9c7a55c62 --verbose
```

When the `atsGuiTLSEnabled` parameter is set to true:

```
curl --request POST https://10.75.217.25:30301/job/SCP-Regression/
buildWithParameters --user
    scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
caroot.cer
```

Starting a Job Forcibly using Custom API

If another job is already running and has not been started by an API user, the running job is aborted, along with all other jobs in the queue that have not been started by an API user, and a new job is started.

If the running job is started by the API user, the new job does not start, and the start job request fails, returning a message in response: `Build <job_id> of pipeline <pipeline_name> is already running, triggered by an API user.`

Builds are aborted gracefully by a forceful API, such as when a running scenario completes its execution and cleanup before the corresponding build is aborted.

The forceful API now returns an `aborted-builds` parameter in response, which contains job IDs for all the aborted builds. It also returns a parameter called `cancelled_builds_in_queue`, which contains queue IDs for all the builds aborted in queue.

If a job ID is assigned to a build in queue, it contains a list of two values: [queueid, jobid] rather than just the queue ID.

Run the following command to start a job forcibly:

```
curl -s --request POST <Startjob_host_port>/build -H "Content-Type:
application/json" -d '{"pipelineName": "<Pipeline_name>", "pageAndQuery":
"<pageAndQuery>"}' --user <username>:<token> --verbose
```

When the `atsGuiTLSEnabled` parameter is set to true:

```
curl -s --request POST
      <Startjob_host_port>/build -H "Content-Type: application/json" -d
'{"pipelineName":  "<Pipeline_name>", "pageAndQuery":
      "<pageAndQuery>"}' --user <username>:<token>   --verbose
      --cacert <path_to_root_certificate>
```

For example,

```
curl --request POST http://10.75.217.25:31170/build  -H "Content-Type:
application/json" -d
    '{"pipelineName": "Policy-Regression",
    "pageAndQuery": "buildWithParameters"}'
    --user policyapiuser:11c1a628f808972c846c510151afa13ba2 --verbose
```

When the `atsGuiTLSEnabled` parameter is set to true:

```
curl --request POST https://10.75.217.25:30170/build  -H "Content-Type:
application/json" -d
    '{"pipelineName": "Policy-Regression",
    "pageAndQuery": "buildWithParameters"}'
    --user policyapiuser:11c1a628f808972c846c510151afa13ba2 --verbose --cacert
    caroot.cer
```

The details of the parameters for the API are as follows:

Table 2-2 API Parameters

Parameters	Mandatory	Default Value	Description
username	YES	NA	This parameter indicates the name of API user.
token	YES	NA	This parameter indicates the API token for API user.
Startjob_host_port	YES	NA	This parameter's format is `<host>:<port>` • `<host>` will be same as Jenkins host • `<port>` will be different(5001 or its nodeport)

Table 2-2 (Cont.) API Parameters

Parameters	Mandatory	Default Value	Description
pipelineName	YES	NA	This parameter indicates the name of the pipeline for which build is to be triggered.
pageAndQuery	YES	NA	This parameter can have two values: • buildWithParameters: for parametrized pipelines • build: for non-parametrized pipelines
jenkins_wait_time	NO	5	This parameter indicates the wait time for Jenkins in seconds. • If Jenkins is very slow in responding and the API response is not as expected, this wait time can be increased. • It is required when multiple running builds must be aborted before starting a new API build. • This value can be provided along with the parameters "pipelineName" and "pageAndQuery" in a similar way. For example: `{"jenkins_wait_time": "10"}`

Customizing Job Parameters

Both of the Start APIs start the pipeline job with default parameter values. You can provide a different value for a parameter in an API call, such as `paramx=valuex`.

1. Append `paramx=value` to `buildWithParameters?`.
   Example 1,

   ```
   curl --request POST 10.75.217.40:31378/job/Policy-NewFeatures/
   buildWithParameters?paramx=valuex --user
         policyapiuser:110ed65222b9e63445689314998ff8c3bk -- verbose
   ```

   Example 2,

   ```
   curl --request POST 10.75.217.4:32476/build -H "Content-Type: application/
   json" -d '{
         "pipelineName": "Policy-NewFeatures", "pageAndQuery":
   "buildWithParameters?paramx=valuex" }'--user <username>:<token>
      --verbose
   ```

2. To add more than 1 parameter, such as `paramx=valuex` and `paramy=valuey`, append the other parameters to the API call using `&`.

Example 1,

```
curl --request POST 10.75.217.40:31378/job/Policy-NewFeatures/
buildWithParameters?paramx=valuex&paramy=valuey
--user policyapiuser:110ed65222b9e63445689314998ff8c3bk -- verbose
```

Example 2,

```
curl --request POST 10.75.217.4:32476/build -H "Content-Type: application/
json" -d '{"pipelineName": "Policy-NewFeatures", "pageAndQuery":
"buildWithParameters?paramx=valuex&paramy=valuey" }' --user
<username>:<token> --verbose
```

3. Replace `buildWithParameters?` with `build` for non-parametrized pipeline jobs.
4. Start the pipeline by using the default Jenkins API or by changing the pageAndQuery parameter's value to build in the following way:

```
curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/build --user
<username>:<API_token> --verbose
```

Example 1,

```
curl --request POST 10.75.217.40:31378/job/Policy-NewFeatures/build --user
policyapiuser:110ed65222b9e63445689314998ff8c3bk --
      verbose
```

Example 2,

```
curl --request POST 10.75.217.4:32476/build -H "Content-Type: application/
json" -d '{
      "pipelineName": "Policy-NewFeatures", "pageAndQuery": "build" }'--
user <username>:<token> –verbose
```

Currently, the ATS API has enhanced capabilities to include triggering builds for running individual or multiple features, scenarios, stages, groups, and tagged executions.

```
curl --request
 POST <IP>:<PORT>/build -H
 "Content-Type: application/json" -d '{"pipelineName": "<NF PIPELINE>",
"pageAndQuery": "buildWithParameters", "otherBuildParameters": { }}' --user
 <NF>apiuser:<token> --verbose
```

When the value of `atsGuiTLSEnabled` parameter is set to true:

```
curl --request POST <IP>:<PORT>/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "<NF PIPELINE>",
    "pageAndQuery": "buildWithParameters",
    "otherBuildParameters": {
    }
  }' \
```

```
--user <NF>apiuser:<Token> \
--verbose
```

> **Note**
>
> The API continues to support the same functionality as in the previous release. In addition, to provide extended support, a new key "otherBuildParameters" has been introduced. This key can be included in the JSON payload sent to the server.

The following table lists the details of "otherBuildParameters" for the API request:

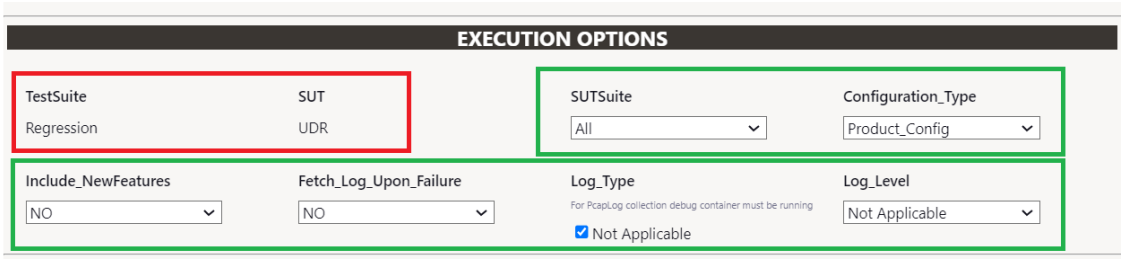
Table 2-3 otherBuildParameters Details

Parameter	Mandatory/Optional	Default Value	Description
otherBuildParameters	Optional	NA	"otherBuildParameters" is a dictionary of build parameters. You can add key-value pairs to customize the build process. For example:`"otherBuildParameters" : { "Features" : "YamlSchema_Import_Export,Custom_Jsons", "Stages" : "stage2,stage3", "Groups" : { "stage1" : "group1,group4", "stage4" : "group4,group5" } }` This dictionary supports the following keys: • Features • Scenarios • FeaturesAndScenarios • Stages • Groups • Feature_Include_Tags • Feature_Exclude_Tags • Scenario_Include_Tags • Scenario_Exclude_Tags

> **Note**
>
> If none of the keys in "otherBuildParameters" are included in the API request, all the test cases with the given execution options will be triggered.

Parameters that are static in the UI should not be included in the "pageAndQuery" parameter of the request. Only parameters with multiple options should be passed. For example, in the UDR pipeline image below, the variables highlighted in red should not be included in the API request, while the parameters highlighted in green can be passed.

Figure 2-6 Execution Option



The following is the format for the API request:

- Example format for executing features, stages, scenarios, or groups through the API request:

```
curl --request POST <IP>:<PORT>/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "<NF PIPELINE>",
    "pageAndQuery": "buildWithParameters",
    "otherBuildParameters": {
      "Features": "<Feature List>",
      "Stages": "<stages>",
      "Groups":{ "<stage-n>":"<group-a,group-b>","<stage-m>":"<group-
p,group-q>" },
      "Scenarios": "",
      "Featuresandscenarios": {
        "<Feature 1>": "<Scenario1>",
        "<Feature2>": "<Scenario1>,<Scenario2>"
      }
    }
  }' \
  --user <NF>apiuser:<Token> \
  --verbose
```

- Example format for executing test cases based on provided tags. If tags are specified, other keys such as "Features", "Scenarios", "Stages", and so on, should not be included.

```
curl --request POST <IP>:<PORT>/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "<NF PIPELINE>",
    "pageAndQuery": "buildWithParameters",
    "otherBuildParameters": {
      "Feature_Include_Tags": "<tags>",
      "Feature_Exclude_Tags": "<tags>",
      "Scenario_Include_Tags": "<Tags>",
      "Scenario_Exclude_Tags": "<Tags>"
    }
  }' \
  --user <NF>apiuser:<Token> \
  --verbose
```

Starting a job with otherBuildParameters

Run Features

In the API request, the user can include all parameters such as SUT, Fetch_Log_Upon_Failure, and so on, within the "pageAndQuery" key. Features can be specified in the request using the "otherBuildParameters" dictionary as shown below:

```
"otherBuildParameters" : { "Features" : "<Comma separated FeatureList>" }
```

For example:

```
curl --request POST 10.75.217.25:30100/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "Policy-Regression",
    "pageAndQuery": "buildWithParameters",
    "otherBuildParameters": {
      "Features":
"NF_Scoring,ManualGetDeleteSession_DeleteSelectivepcfBindingswithSUPI",
    }
  }' \
  --user policyapiuser:11c1a628f808972c846c510151afa13ba2 \
  --verbose
```

Run Scenarios

- Using "Scenarios" key
  In the API request, the user can include all parameters such as SUT, Fetch_Log_Upon_Failure, and so on, under the "pageAndQuery" key. Scenarios can be specified in the request using the "otherBuildParameters" dictionary as follows:

  ```
  "otherBuildParameters" : { "Scenarios" : "<Comma separated Scenarios
  List>" }
  ```

  For example:

  ```
  curl --request POST 10.75.217.25:30100/build \
    -H "Content-Type: application/json" \
    -d '{
      "pipelineName": "Policy-Regression",
      "pageAndQuery": "buildWithParameters",
      "otherBuildParameters": {
        "Scenarios":
  "Re_Import_YamlSchema_Verify_SMPolicy,AM_Terminate_Notify_Timeout,AM_Notify
  _With_Header_Timeout"
      }
    }' \
    --user policyapiuser:11c1a628f808972c846c510151afa13ba2 \
    --verbose
  ```

- Using "Featuresandscenarios" key

The API request allows the user to include all parameters such as SUT, Fetch_Log_Upon_Failure, and so on, using the "pageAndQuery" key. Scenarios can be specified within the "otherBuildParameters" dictionary, as follows:

```
"otherBuildParameters" : { "Featuresandscenarios" : { "<Feature -1>" :
"<Comma separated scenarios from Feature -1>" , "Feature -2" : "<Comma
separated scenarios from Feature -2>" } }
```

For example:

```
curl --request POST 10.75.217.25:30100/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "Policy-Regression",
    "pageAndQuery": "buildWithParameters",
    "otherBuildParameters": {
      "Featuresandscenarios":{
         "YamlSchema_Import_Export" :
"Re_Import_YamlSchema_Verify_SMPolicy"
       }
    }
  }' \
  --user policyapiuser:11c1a628f808972c846c510151afa13ba2 \
  --verbose
```

Run Stages

In the API request, the user can provide all parameters such as SUT, Fetch_Log_Upon_Failure, etc., using the "pageAndQuery" key. Stages can be specified within the "otherBuildParameters" dictionary, as shown below:

```
"otherBuildParameters" : { "Stages" : "<Comma separated Stages list>" }
```

For example:

```
curl --request POST 10.75.217.25:30100/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "Policy-Regression",
    "pageAndQuery": "buildWithParameters?Configuration_Type=Custom_Config",
    "otherBuildParameters": {
      "Stages" : "stage2,stage4"
    }
  }' \
  --user policyapiuser:11c1a628f808972c846c510151afa13ba2 \
  --verbose
```

Run Groups

In the API request, the user can include all parameters such as SUT, Fetch_Log_Upon_Failure, and so on, under the "pageAndQuery" key. Groups can be specified within the "otherBuildParameters" dictionary, as follows:

```
"otherBuildParameters" : { "Groups" : {  "<stage n>" : "<Comma
    separated Groups List from stage n>", "<stage m>" : "<Comma  separated
Group list from
    stage m>"  } }
```

For example:

```
curl --request POST 10.75.217.25:30100/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "Policy-Regression",
    "pageAndQuery": "buildWithParameters",
    "otherBuildParameters": {
      "Groups":{"stage1":"group1" , "stage2" : "group3,group6"}
    }
  }' \
  --user policyapiuser:11c1a628f808972c846c510151afa13ba2 \
  --verbosee
```

Run with Tags

In the API request, you can specify all parameters such as SUT, Fetch_Log_Upon_Failure, and so on, using the "pageAndQuery" key. When using tags, it is mandatory to include "FilterWithTags" in the "pageAndQuery". If tags are provided by you, and if any feature or scenario is provided, they are not considered. Only tags and stages or groups are provided as part of the request.

Tags can be specified in the request within the "otherBuildParameters" dictionary as follows:

```
"otherBuildParameters": {  "Feature_Include_Tags":"<tags>",
"Feature_Exclude_Tags":"<tags>", "Scenario_Include_Tags":"<Tags>" ,
"Scenario_Exclude_Tags":"<Tags>" }
```

> **Note**
>
> The provided tags must be separated by commas.

For example:

```
curl --request POST 10.75.217.25:30100/build \
  -H "Content-Type: application/json" \
  -d '{
    "pipelineName": "Policy-Regression",
    "pageAndQuery": "buildWithParameters?FilterWithTags=Yes",
    "otherBuildParameters": {
      "Feature_Include_Tags":"cne-common,cm-service",
"Scenario_Include_Tags":"cleanup,sanity"
    }
  }' \
```

```
--user policyapiuser:11c1a628f808972c846c510151afa13ba2 \
--verbose
```

> **Note**
>
> The API behaves in a manner similar to that of the UI. For instance, when running a set of scenarios, the user selects them, and only those chosen are run upon triggering the build. If any stages, groups, or features are also selected in the features section, they are ignored. Similarly, if stages, groups, or features are included alongside scenarios in the API request, only the scenarios will be run.

2.1.2.3 Monitoring Jobs

This Default Jenkins API is used to monitor the progress of the job that was started.

For monitoring, the following APIs are used:

- A `qid` is obtained from the Location header in the response for `starting a job`. The first API uses this `qid` to get queue status about the corresponding job, including its `job_id`.
- The second API uses the `job_id` to obtain further information about the job status.

Monitoring a Job
To monitor jobs, run the following commands in a sequence:

1. `curl --request POST <Jenkins_host_port>/queue/item/<qid>/api/json --user <username>:<API_token> --verbose`
   When the `atsGuiTLSEnabled` parameter is set to true:

   ```
   curl --request POST
         <Jenkins_host_port>/queue/item/<qid>/api/json --user
   <username>:<API_token> --verbose --cacert
         <path_to_root_certificate>
   ```

   For example,

   ```
   curl --request POST http://10.123.154.163:30427/queue/item/5/api/json--
   user policyapiuser:111ad02d7471cec9ca689696e9c7a55c62
            --verbose
   ```

   When the `atsGuiTLSEnabled` parameter is set to true:

   ```
   curl --request POST https://10.75.217.25:30301/queue/item/27/api/json --
   user
         scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
       caroot.cer
   ```

2. `curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/api/json --user <username>:<API_token> --verbose`
   When the `atsGuiTLSEnabled` parameter is set to true:

   ```
   curl --request POST
         <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/api/json  --user
   ```

```
<username>:<API_token> --verbose
      --cacert<path_to_root_certificate>
```

For example,

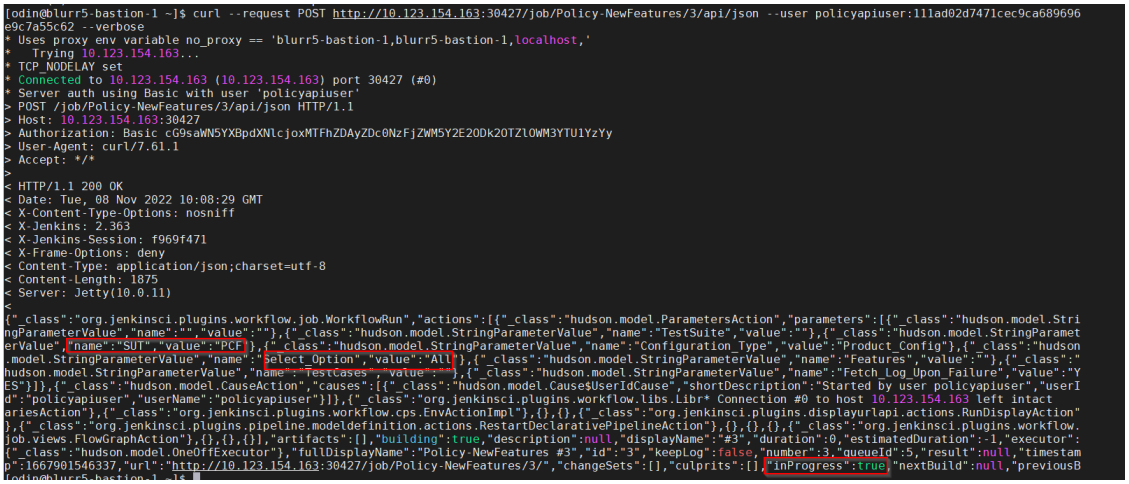
```
curl --request POST http://10.123.154.163:30427/job/Policy-
NewFeatures/3/api/json--user
policyapiuser:111ad02d7471cec9ca689696e9c7a55c62
          --verbose
```

When the `atsGuiTLSEnabled` parameter is set to true:

```
curl --request POST https://10.75.217.25:30301/job/SCP-Regression/2/api/
json --user
      scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
    caroot.cer
```

The following screenshot shows an example of monitoring the progress of the job:

Figure 2-7 Monitoring a Job



2.1.2.4 Stopping Jobs

Stop API is used to stop the currently running job using its job_id. It is also a default Jenkins API.

Stopping a Job

For ATS without Parallel Test Execution framework integrated:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/stop  --user
<username>:<API_token>
    --verbose
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/stop  --user
<username>:<API_token> --verbose –cacert
      <path_to_root_certificate>
```

For example,

```
curl --request POST http://10.75.217.4:31881/job/UDR-Regression/21/stop --user
    udrapiuser:1139a72213e0a686972cbff4a2f9333a9f --verbose
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST https://10.75.217.25:30301/job/SCP-Regression/2/stop --
user
    scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
caroot.cer
```

Note

- If the rerun count is greater than zero, the job must be stopped twice.
- This Stop API call does not abort the build gracefully.

For ATS with Parallel Test Execution framework integrated:

```
curl --request POST
      <Stopjob_host_port>/job/<Pipeline_name>/<job_id>/stop  --user
<username>:<API_token>
    --verbose
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST
      <Stopjob_host_port>/job/<Pipeline_name>/<job_id>/stop  --user
<username>:<API_token> --verbose --cacert
      <path_to_root_certificate>
```

For example.

```
curl --request POST http://10.75.217.4:32476/job/UDR-Regression/21/stop --user
    udrapiuser:1139a72213e0a686972cbff4a2f9333a9f --verbose
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST https://10.75.217.25:30170/job/SCP-Regression/2/stop --
user
    scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
caroot.cer
```

The following table lists the parameter details for Stop API:

Table 2-4 Stop API Details

Parameter	Mandatory	Default Value	Description
userName	Yes	NA	Name of API user
token	Yes	NA	The API token for the API user
Stopjob_host_port	Yes	NA	Format is <host>:<port> • <host> is same as Jenkins host • <port> is different such as 5001 or its nodeport
pipelineName	Yes	NA	Name of the pipeline for which build is to be stopped
immediate	No	False	To stop the build immediately, send a query parameter ("immediate=true") with API call. For example, `curl --request POST <Stopjob_host_port>/job/<Pipeline_name>/<job_id>/stop?immediate=true --user <username>:<API_token> --verbose` • `immediate` can also have values such as yes or 1. These values work similar to the true value.

2.1.2.5 Getting Test Suite Artifacts

Default Jenkins API is used to get the JUNIT-formatted XML test result files for a completed test suite.

For getting artifacts for any completed test suite, the following APIs are used:

- For getting an overall build summary
- For getting a JUNIT XML test result file for every feature file that ran

For getting an overall build summary:

```
curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/
testReport/api/xml?exclude=testResult/suite --user <username>:<API_token> --
verbose
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/testReport/api/xml?
exclude=testResult/suite  --user
      <username>:<API_token> --verbose --cacert  <path_to_root_certificate>
```

For example,

```
curl --request POST http://10.123.154.163:30427/job/Policy-NewFeatures/4/
testReport/api/xml?exclude=testResult/suite--user
policyapiuser:111ad02d7471cec9ca689696e9c7a55c62 --verbose
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST https://10.75.217.25:30301/job/SCP-Regression/1/
testReport/api/xml?exclude=testResult/suite
      --user scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
    caroot.cer
```

For getting Feature-wise XML, Select_Option = All:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/artifact/test-results/
reports/*zip*/test-results.zip
      --user <username>:<API_token> --verbose --output  test-results.zip
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/artifact/test-results/
reports/*zip*/test-results.zip
      --user <username>:<API_token> --verbose --output  test-results.zip --
cacert
      <path_to_root_certificate>
```

For example,

```
curl --request POST http://
        10.75.217.4:31881/job/Policy-NewFeatures/21/artifact/test-results/
reports/*zip*/test-results.zip  --user
policyapiuser:11c3344996c4fda01ded2124bec4f9aa17
      --verbose –-output test-results.zip
```

When the atsGuiTLSEnabled parameter is set to true:

```
curl --request POST https://10.75.217.25:30301/job/SCP-Regression/1/artifact/
test-results/reports/*zip*/test-results.zip
    --user scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
    caroot.cer
```

For getting Feature-wise XML, Select_Option = Single/MultipleFeatures:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/artifact/test-results/
reports/
      *.<Feature1_name>.xml,*.<Feature2_name>.xml/*zip*/test-results.zip  --
user <username>:<API_token>
      --verbose --output  test-results.zip
```

When the `atsGuiTLSEnabled` parameter is set to true:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/artifact/test-results/
reports/
      *.<Feature1_name>.xml,*.<Feature2_name>.xml/*zip*/test-results.zip  --
user <username>:<API_token>
      --verbose --output  test-results.zip –-cacert
    <path_to_root_certificate>
```

For example,

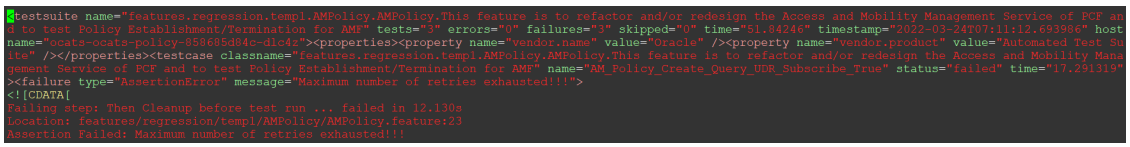
```
curl --request POST http://
      10.75.217.4:31881/job/Policy-NewFeatures/21/artifact/test-results/
reports/*.goldenfeature.xml,*.AMPolicy.xml/*zip*/test-results.zip
      –user policyapiuser:11c3344996c4fda01ded2124bec4f9aa17 --verbose  --
output test-results.zip
```

When the `atsGuiTLSEnabled` parameter is set to true:

```
curl --request POST https://10.75.217.25:30301/job/SCP-Regression/1/artifact/
test-results/reports/*.SCP_Registration_With_PLMNList.xml/*zip*/test-
results.zip
    --user scpapiuser:11c2fde49cea6eb8f332ad23a7877ea2de --verbose --cacert
    caroot.cer
```

API calls for `Select_Option = All and Select_Option = Single/MultipleFeatures` return a zip file with JUNIT XMLs, one XML for each feature.

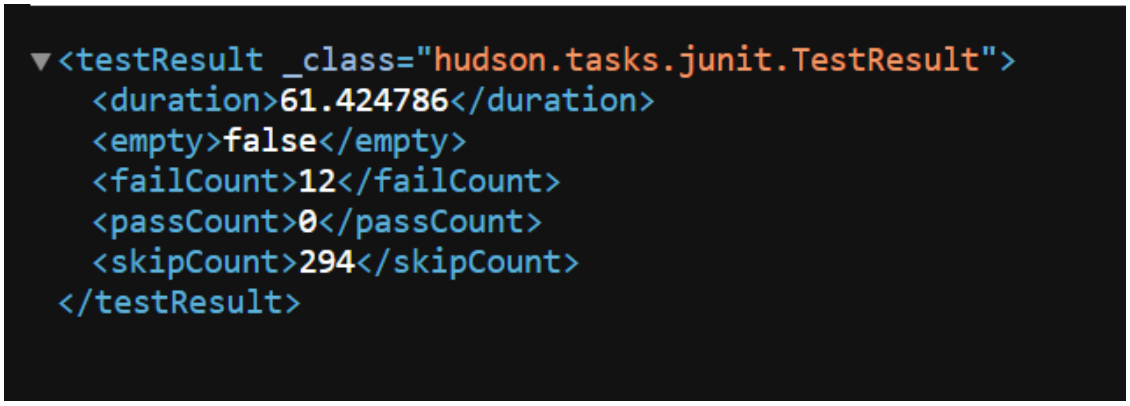
Figure 2-8 Sample XML Output for AMPolicy.feature



In the API call, specify other selected features in comma-separated form as `/*<Feature1_name>.xml,*<Feature2_name>.xml,*<Feature3_name>.xml,*<Feature4_name>.xml/` for Select_Option = Single/MultipleFeatures.

The API call for getting the overall build summary returns an XML with values for `duration`, `failCount`, `passCount`, and `skipCount` for the current build.

Figure 2-9 Sample Output



It is recommended to maintain a gap of at least a few seconds between two API calls. This gap depends on the time Jenkins takes to complete the API request.

2.2 ATS Custom Abort

ATS Custom Abort feature allows you to gracefully abort the ongoing build directly from the Graphical User Interface (GUI).

You can abort the builds in ATS using the following ways:

- **Using the Abort Button on the GUI**: This method, supported by Jenkins, allows you to abort builds directly from the user interface.
- **Using the ATS API**: This is a partially manual method for aborting builds.
  When the ATS API is used to abort a build, ATS will wait for any running scenarios to complete their cleanup process before finalizing the abort. This ensures that there are no issues related to cleanup when using the ATS API.

Manual Abort

By default, Jenkins provides a **cancel** icon for every pipeline on the dashboard whenever a job is running. However, using the manual abort or **cancel** icon has some limitations, especially in parallel execution scenarios.

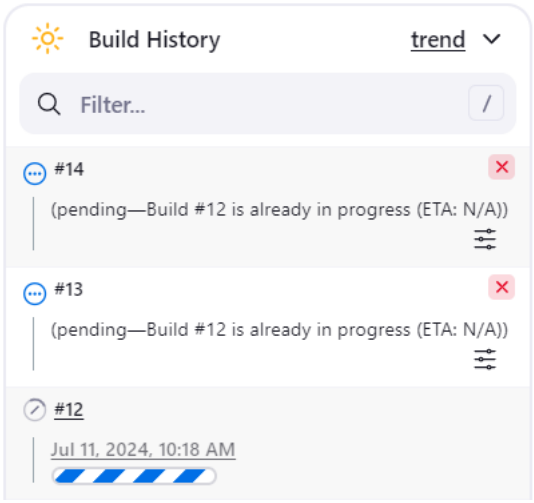
When the manual abort or **cancel** icon is used, Jenkins sends a kill signal to stop all current executions, regardless of whether test cases or other operations are in progress. In cases of parallel execution with multiple stages, the manual abort or **cancel** icon may need to be clicked multiple times due to the presence of multiple stages. This approach can lead to pending cleanup for test cases, which may cause failures in subsequent executions.

To address the issues with manual abort, the release 24.3.0 introduces an **Abort_Build** menu in all new features and regression pipelines, ensuring a more controlled and graceful termination of builds.

The **cancel** icon will not be displayed for currently running builds. However, if there are any builds in the queue, they will still have the **cancel** icon available for use.

For example, builds 13 and 14 can be stopped using the **cancel** icon next to them, while build 12 can only be stopped using the new **Abort_Build** in the left navigation pane.

Figure 2-10 Stopping Builds



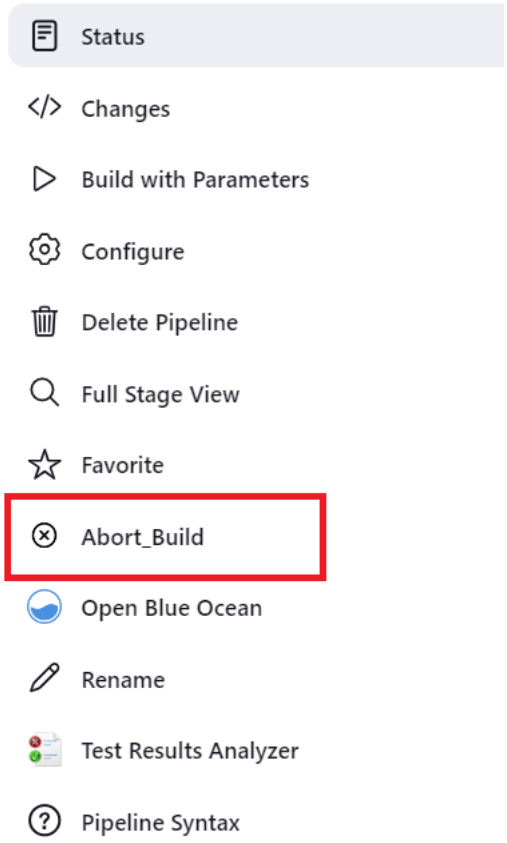
Abort_Build

The **Abort_Build** is available on every regression and new features pipeline. It supports the graceful termination of builds. This **Abort** button available in the GUI triggers a stop API request to the ATS API server. This request ensures that the execution is stopped gracefully, allowing all currently running scenarios to complete.

Using Abort Button

1. Enter the login credentials and click **Sign in**.
   The screens displays preconfigured pipelines for NF individually.
2. Click **NF-NewFeatures** or **NF-Regression** in the **Name** column.
   The NF-NewFeatures or NF-Regression screen appears.
3. Click **Abort_Build** in the left navigation pane.

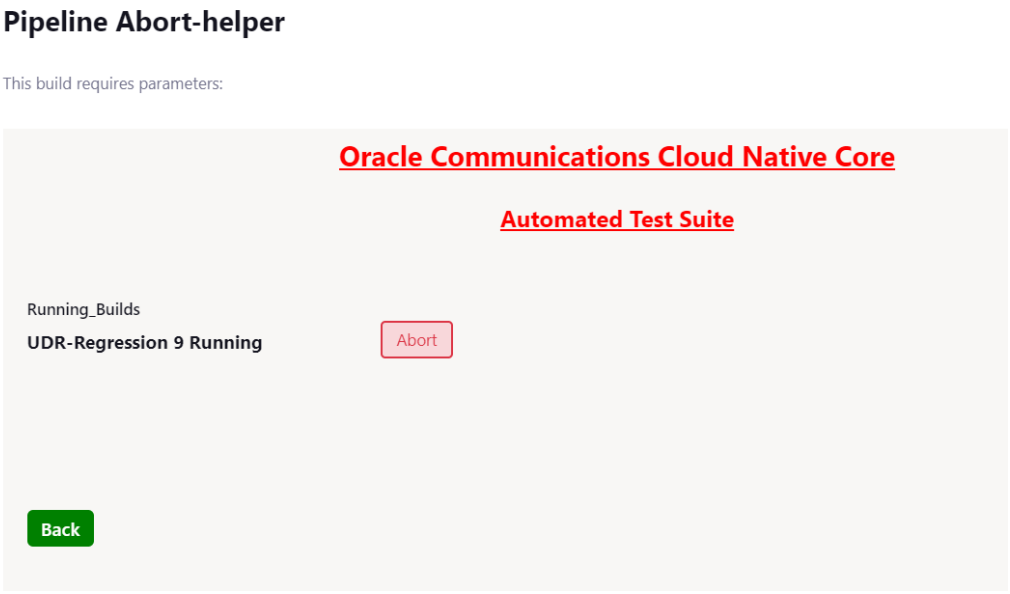
Figure 2-11 Abort Build



You will be redirected to the Pipeline Abort-helper page.

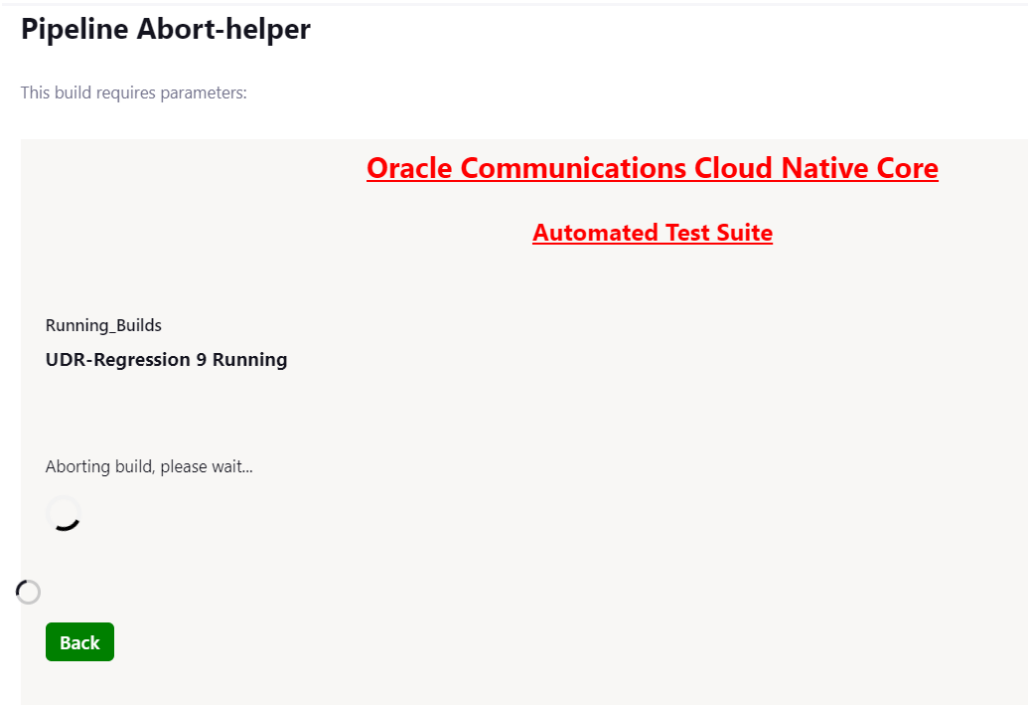
4. If no builds are currently running or in progress, the page will display the message: "No builds are running."
5. If builds are running, the page will display the following:
   - **Running_Builds**: This section lists the running pipeline names and build numbers.

Figure 2-12 Running_Builds



- **Abort**: Click the button to start the abort process.

Figure 2-13 Aborting Build



- **Back**: Click the button to return to the pipeline page or remain on the current page while the abort process completes.

Time Taken by Custom Abort to Stop a Build

The time required to stop a build gracefully depends on various factors, including the current stage of the build:

- If the abort is initiated during stages such as "Preparation" or "POST," the build stops within a few minutes.
- If parallel test case execution is not enabled for NFs, and there is only one stage, such as "Execute-tests" or "stage1/group1", the time taken for a graceful abort depends on the duration of the currently running scenarios.
- If parallel test case execution is enabled for NFs, the time required for abort depends on the scenario that takes the longest time to complete among the currently running groups.

On the Console Output page of any running build, for example, http://<IP>:<Node port>/job/<Job Name>/<Build Number>/console, Jenkins will displays a manual abort or **cancel** icon.

Figure 2-14 Manual Abort or Cancel icon

> **Note**
>
> It is recommended to use the new **Abort** button instead of the manual abort or **cancel** icon provided by Jenkins for a more reliable abort process.

2.3 ATS Feature Activation and Deactivation

ATS Feature Activation and Deactivation feature allows users to activate or deactivate specific features within the ATS using Helm charts.

> **Note**
>
> When these features are removed, they cannot be reinstated in the deployed ATS. However, users have the option to reinstall the ATS to restore the disabled features.

The following table lists the features to enable or disable using Helm charts.

> **Note**
>
> These parameters can be edited in the ATS deployment file (`values.yaml`).

Table 2-5 Enable or Disable ATS Feature

Features	Parameter	Description
Support for Test Case Mapping and Count	`testCaseMapping`	Set this parameter to `true` to activate the feature in the ATS GUI.
Application Log Collection and PCAP Log Collection	`logging`	Set this parameter to `true` to collect the ATS logs. If the parameter is set to false, the logs will not be collected.

Table 2-5 (Cont.) Enable or Disable ATS Feature

Features	Parameter	Description
Lightweight Performance	`lightWeightPerformance`	Set this parameter to `true` to activate the feature in the ATS GUI. If the parameter is set to false, the performance pipeline will not be accessible.
ATS Health Check	`healthcheck`	Set this parameter to `true` to activate the feature in the ATS GUI. If the parameter is set to false, the health check pipeline will not be accessible.
ATS API	`atsApi`	Set this parameter to `true` to activate the feature. If the parameter is set to false, the ATS API feature on the 5001 port will be disabled.
Parameterization	`parameterization`	Set this parameter to `true` to activate the feature in the ATS GUI. If the parameter is set to false, the Configuration_Type parameter on the GUI will not be available.
Parallel Test Execution	`parallelFrameworkChangesIntegrated`	Set this parameter to `true` if all parallel test case execution features are picked by NF and changes are made to files. **Note**: Do not change the values provided in the `values.yaml` file for this parameter.
Parallel Test Execution	`parallelTestCaseExecution`	Set this parameter to `true` to activate the feature in the ATS GUI. If set to false, all the features will be copied into a single stage or group, resulting in sequential execution. **Note**: It is not advisable to edit the default value given in the values.yaml file for this parameter.
Parallel Test Execution	`mergedExecution`	Set this parameter to `true` to activate the feature in the ATS GUI. If set to false, the option to include other pipelines for mergedExecution will not be available. **Note**: It is not advisable to edit the default value given in the values.yaml file for this parameter.
ATS Support to Execute Scenarios	`scenarioSelection`	Set this parameter to `true` to activate the feature in the ATS GUI. If set to false, the ability to select single or multiple scenarios will be removed. It is dependent on the testCaseMapping parameter. If this parameter is set to false, the scenarioSelection parameter will be set to false too. **Note**: It is not advisable to edit the default value given in the values.yaml file for this parameter.

Table 2-5 (Cont.) Enable or Disable ATS Feature

Features	Parameter	Description
ATS Tagging Support	`executionWithTagging`	Set this parameter to `true` to activate the feature in the ATS GUI. If set to false, the ability to execute test cases based on tags will be removed. It is dependent on the testCaseMapping parameter. If this parameter is set to false, the executionWithTagging parameter will be set to false too.
Stage or Group Level Execution	`individualStageGroupSelection`	Set this parameter to `true` to activate the feature in the ATS GUI. If set to false, the ability to select all test cases from individual stages or groups using a single checkbox will be removed. **Note**: It is not advisable to edit the default value given in the values.yaml file for this parameter.
Support for Transport Layer Security	`atsGUITLSEnabled`	Set this parameter to `true` to activate the feature in the ATS GUI. If set to false, ATS will work in HTTP mode.
Prometheus Requires Token-based Authentication ATS connects to Prometheus to retrieve metrics and alerts for validating traffic and configurations during automated testing. Prometheus continuously collects metrics from NFs' services as a central platform for accessing metrics and alerts. You can configure ATS to enable or disable authentication and TLS for communication with Prometheus. Authentication can be enabled if Prometheus requires Role-Based Access Control (RBAC), utilizing a service account token for authorization. ATS uses the token located at `/var/run/secrets/kubernetes.io/serviceaccount/token`, which is included as a bearer token in each request to Prometheus. Kubernetes manages the token, ensuring that any expired tokens are automatically updated within the pod.	`PrometheusAuthEnabled`	Set this parameter to `true` when Prometheus requests require a token in the header for authentication.

Table 2-5 (Cont.) Enable or Disable ATS Feature

Features	Parameter	Description
	PrometheusTLSEnabled	Set this parameter to true when Prometheus has HTTPS deployment. **Note:** When the connection is established using HTTPS, it remains insecure because no certificates are maintained with ATS for this connection. Sample configuration: #The below section is related to the deployments where prometheus needs authenticated token to request for metrics/alerts PrometheusAuthEnabled: false #Set this to true only when prometheus reqeusts requrire a token in the header for authentication PrometheusTLSEnabled: false #Set this to true only when prometheus is a https deployment. #PrometheusScrapeInterval: "60" #Uncomment this line only when scrapeInterval is any other value than 60 seconds.(ATS will assume 60s as default if no response comes from prometheus config.) # Optionally override ClusterRole rules used when PrometheusAuthEnabled is true. prometheusClusterRole: # If rules is empty or this section is omitted, the default rule is used: # - apiGroups: [""] # resources: ["namespaces"] # verbs: ["get"] # - apiGroups: ["monitoring.coreos.com"] # resources: ["prometheuses/api"] # resourceNames: ["k8s"]

Table 2-5 (Cont.) Enable or Disable ATS Feature

Features	Parameter	Description

		`# verbs: ["get", "create", "update"]`
		`# - nonResourceURLs: ["/*"]`
		`# verbs: ["get"]`
		`rules: []`
		`# Example sample rules:`
		`# rules:`
		`# - apiGroups: [""]`
		`# resources: ["pods", "services"]`
		`# verbs: ["get", "list"]`
		`# - nonResourceURLs:`
		`# - "/*"`
		`# verbs:`
		`# - get`

Support for Dual Stack	`deploymentMode`	Set this parameter to one of the possible values to enable the dual stack mechanism. The dual stack mechanism enables ATS to establish connections with REST APIs and ATS GUI in a Kubernetes cluster using IPv4 or IPv6 or both simultaneously. Default value: ClusterPreferred Possible values: IPv4, IPv6, IPv4_IPv6, IPv6_IPv4, and ClusterPreferred. **Note:** This parameter is supported only on Kubernetes 1.23.0 and later.

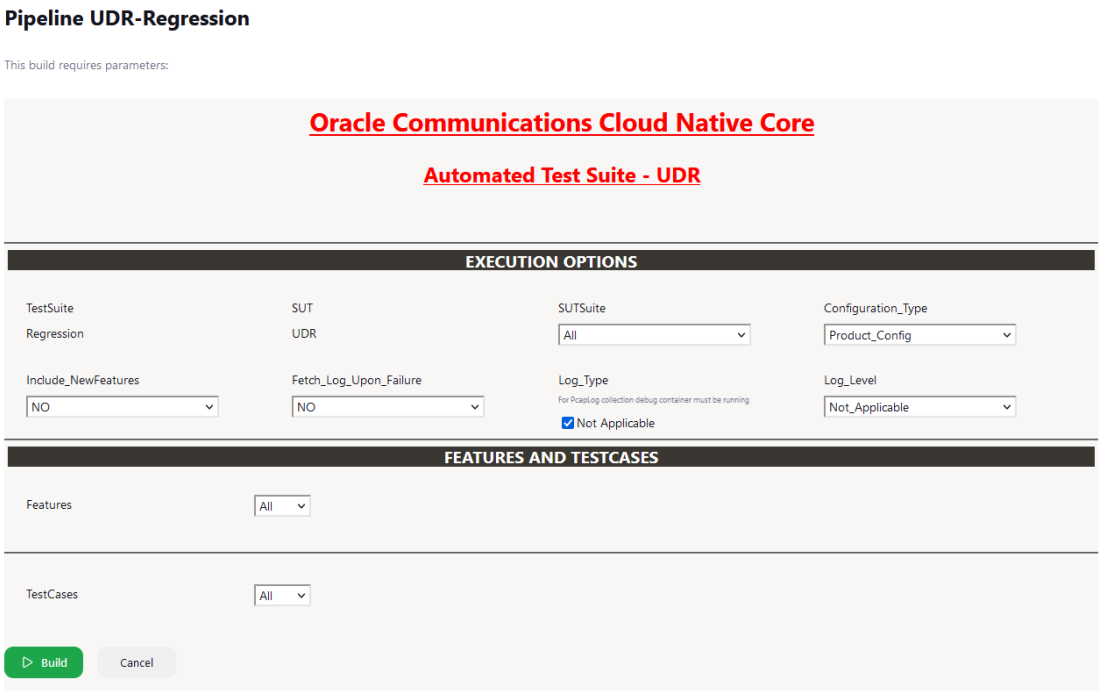
Note

- You can edit the parameters relating to the features that NF supports. Keep the default value for the remaining parameters.
- For the current release, mergedExecution, individualStageGroupSelection, and parallelTestCaseExecution parameters value should not be modified.

2.4 ATS GUI Enhancements

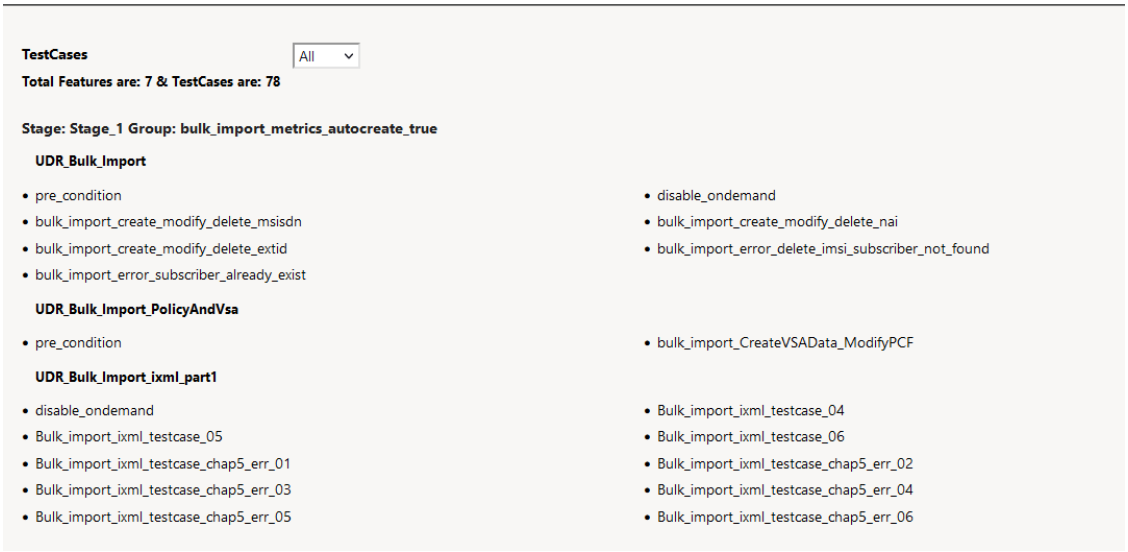
With this enhancement, the ATS Graphical User Interface (GUI) layout is redesigned to streamline user interaction by segregating execution options and features or test cases into distinct compartments, allowing the users to navigate effortlessly between different functionalities.

Figure 2-15 Layout Enhancement



Test cases now feature a display of fixed length, ensuring that even lengthier test cases remain visually manageable. Additionally, the inclusion of hover-scroll functionality allows for easy access to content beyond the visible area, enhancing readability and user friendliness.

Figure 2-16 Test Cases Visibility

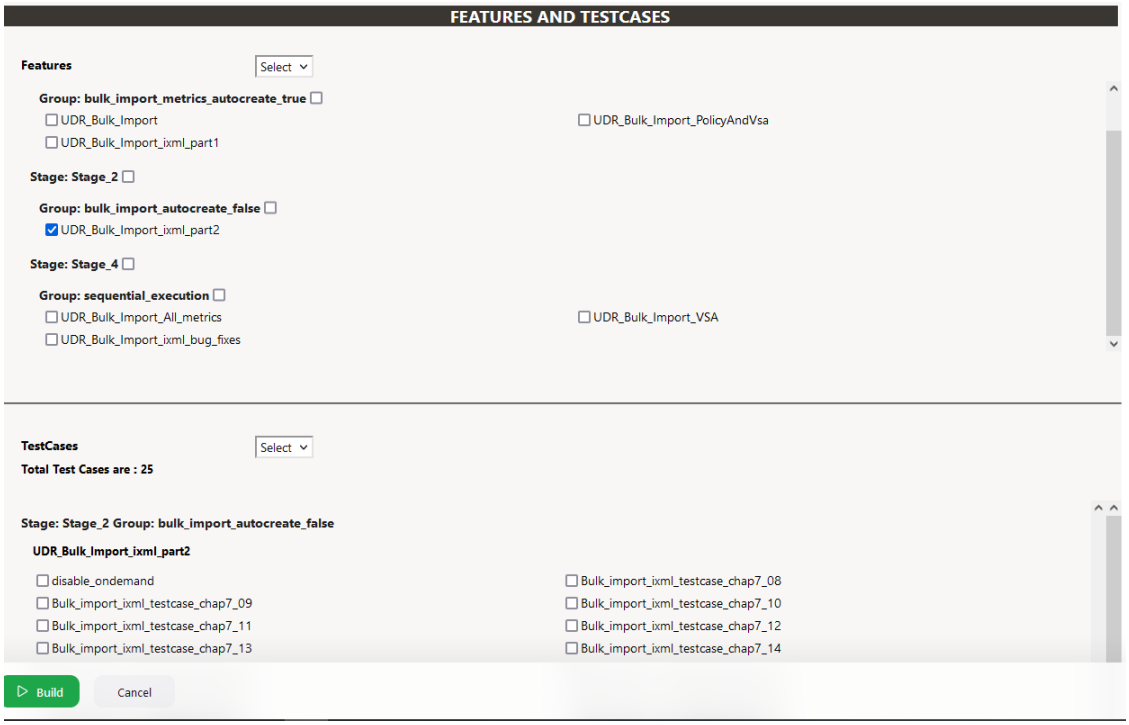


Tooltips have been introduced to allow user to have quick insights into the grouping and staging of individual test cases, thereby improving overall understanding and navigation within the ATS environment.

Figure 2-17 Tooltips

To facilitate the use of multiselect features and scenarios, distinct drop down menus have been introduced. This enables users to efficiently choose and manage multiple options simultaneously, thereby improving flexibility and productivity in selecting test cases.

Figure 2-18 Multiselect Features and Scenarios



2.5 ATS Health Check

The ATS Health Check feature allows you to evaluate the health of the ATS deployment by conducting a comprehensive series of checks. ATS health checks are performed using the

Health Check tool. After installation, it ensures the health of CNCATS pods, their services, and associated configurations.

Overview of Health Check Tool Functionality

The following provides a summary of its main functions:

1. Initial Setup: After installation, the Health Check tool begins by running a series of predefined tests to establish a baseline.
2. CPU and Memory Verification: The Health check tool verifies whether the CPU and memory allocated to CNCATS pods meet the minimum requirements set. It compares current resource allocations with these thresholds and flags any shortfalls, recommending necessary adjustments.
3. Service Verification: It checks the operational status of the CNCATS services (ATS API and ATS GUI), including verifying service endpoints, ensuring services are running, and confirming they respond as expected.
4. Test Folder Validation: It inspects the test folder to ensure that all necessary test files are present and properly configured.
5. Configuration Checks: The tool reviews the authentication configurations required for running the System Under Test (SUT/NF) health check, verifying that all configurations are correct to facilitate smooth run.
6. PVC Verification: It confirms whether the Persistent Volume Claim (PVC) is in a bound state and properly connected to a Persistent Volume. Any issues with PVC binding are flagged for further investigation.

By performing these checks, the tool ensures that the CNCATS pod and its associated services are functioning correctly, identifying potential issues before they affect system performance.

> **Note**
>
> - This feature is available starting from CNCATS 24.3.0 Build.
> - For initial checks, view the ATS Health Check in pod logs using the command:
>
>   ```
>   kubectl logs <podname> -n <namespace>
>   ```
>
>   .
> - For subsequent checks, rerun the tool through ATS bash with the command:
>
>   ```
>   healthtest
>   ```
>
>   .

The following screenshots provide visual examples of different scenarios:

Figure 2-19 Success Health Check

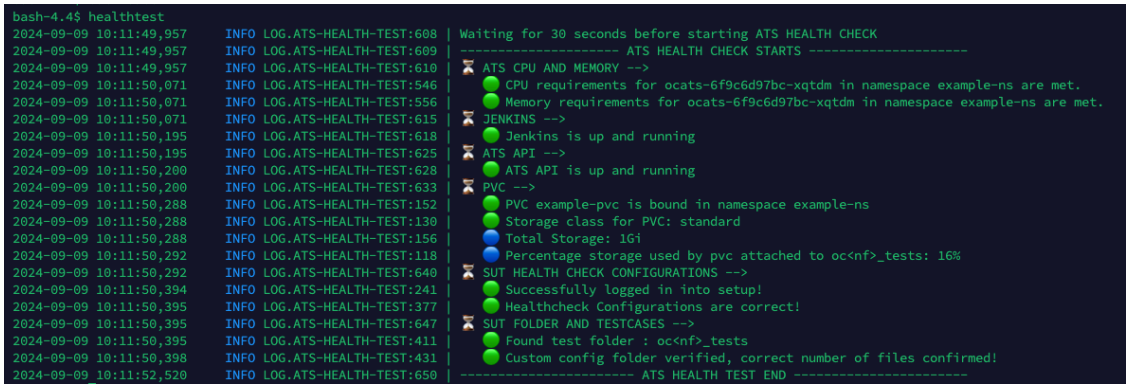
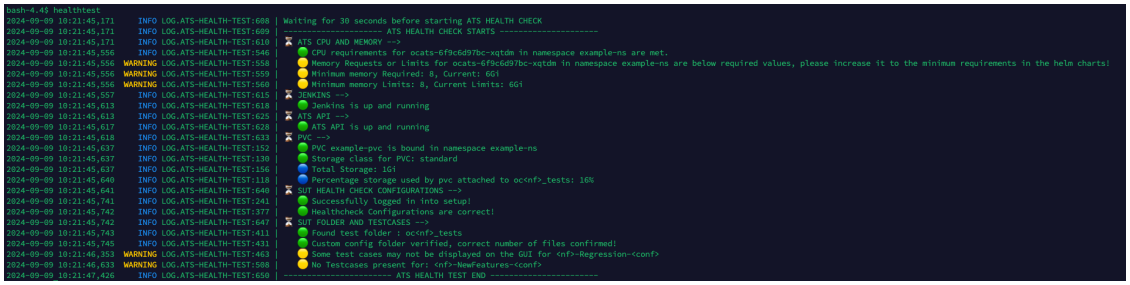


Figure 2-20 Warning and Errors



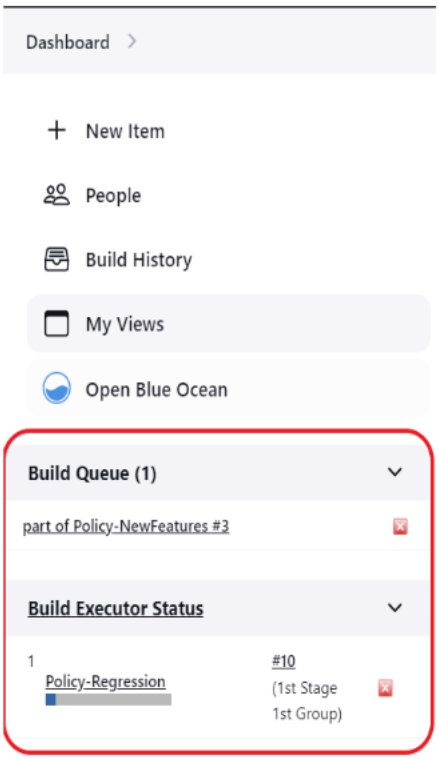
> **Note**
>
> The highlighted areas illustrate a health check with warnings and errors that require further investigation or action.

2.6 ATS Jenkins Job Queue

The ATS Jenkins Job Queue feature places the second job in a queue if the current job is already running from the same or different pipelines to prevent jobs from running in parallel to one another.

Job/build queue status can be viewed in the left navigation pane on the ATS home page. The following image shows the build queue status when a user has tried to run the NewFeatures pipeline when the Regression pipeline is already running.

Figure 2-21 Build Executor Status



2.7 Application Log Collection

Using Application Log Collection, you can debug a failed test case by collecting the application logs for NF System Under Test (SUT). Application logs are collected for the duration that the failed test case was run.

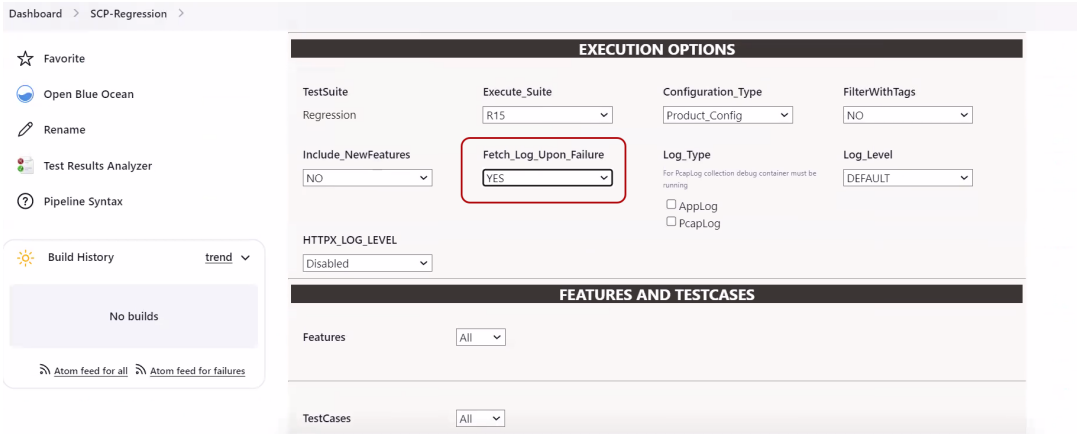
Application Log Collection can be implemented by using OpenSearch or Kubernetes Logs. In both these implementations, logs are collected per scenario for the failed scenarios.

Application Log Collection Using OpenSearch

To access the option to collect logs using OpenSearch:

1. Log in to ATS using respective <NF> login credentials.
2. On the NF home page, click any new feature or regression pipeline, from where you want to collect the logs.
3. In the left navigation pane, click **Build with Parameters**.
4. Select YES or NO from the drop-down menu of **Fetch_Log_Upon_Failure** to select whether the log collection is required for a particular run.

Figure 2-22 Fetch_Log_Upon_Failure



5. If option **Log_Type** is also available, select value **AppLog** for it.
6. Select the Log Level from the drop-down menu of Log_Level to set the log level for all the microservices. The possible values for Log_Level are as follows:
   - WARN: Designates potentially harmful situations.
   - INFO: Designates informational messages that highlight the progress of the application at coarse-grained level.
   - DEBUG: Designates fine-grained informational events that are most useful to debug an application.
   - ERROR: Designates error events that might still allow the application to continue running.
   - TRACE: The TRACE log level captures all the details about the behavior of the application. It is mostly diagnostic and is more granular and finer than DEBUG log level.

   > **Note**
   >
   > Log_Level values are NF dependent.

7. After the build execution is complete, go into the ATS pod, then navigate to following path to find the applogs:`.jenkins/jobs/<Pipeline Name>/builds/<build number>/` For example,`.jenkins/jobs/SCP-Regression/builds/5/`

   Applogs is present in zip form. Unzip it to get the log files.

The following tasks are carried out in the background to collect logs:

- OpenSearch API is used to access and fetch logs.
- Logs are fetched from OpenSearch for the failed scenarios
- Hooks (after scenario) within the cleanup file initiate an API call to OpenSearch to fetch Application logs.
- Duration of the failed scenario is calculated based on the time stamp and passed as a parameter to fetch the logs from OpenSearch.
- Filtered query is used to fetch the records based on Pod name, Service name, and timestamp (Failed Scenario Duration).

- For OpenSearch, there is no rollover or rotation of logs over time.
- The following configuration parameters are used for collecting logs using OpenSearch:
  - OPENSEARCH_WAIT_TIME: Wait time to connect to OpenSearch
  - OPENSEARCH_HOST: OpenSearch HostName
  - OPENSEARCH_PORT: OpenSearch Port

Application Log Collection Using Kubernetes Logs

To access the option to to collect logs using Kubernetes Logs:

1. On the NF home page, click any new feature or regression pipeline, from where you want to collect the logs.
2. In the left navigation pane, click **Build with Parameters**.
3. Select YES or NO from the drop-down menu of **Fetch_Log_Upon_Failure** to select whether the log collection is required for a particular run.
4. Select the Log Level from the drop-down menu of Log_Level to set the log level for all the microservices. The possible values for Log_Level are as follows:
   - WARN: Designates potentially harmful situations.
   - INFO: Designates informational messages that highlight the progress of the application at coarse-grained level.
   - DEBUG: Designates fine-grained informational events that are most useful to debug an application.
   - ERROR: Designates error events that might still allow the application to continue running.

> **Note**
>
> Log_Level values are NF dependent.

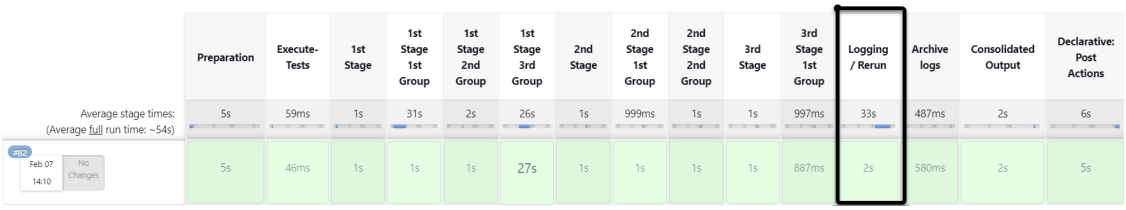
The following tasks are carried out in the background to collect logs:

- Kube API is used to access and fetch logs.
- For failed scenarios, logs are directly fetched from microservices.
- Hooks (after scenario) within the cleanup file initiate an API call to Kubernetes Logs to fetch Application logs.
- The duration of the failed scenario is calculated based on the time stamp and passed as a parameter to fetch the logs from microservices.
- Logs roll can occur while fetching the logs for a failed scenario. The maximum loss of logs is confined to a single scenario.

2.7.1 Application Log Collection and Parallel Test Execution Integration

A new stage,"Logging/Rerun", has been added at the end of the Execute-Tests stage to collect rerun logs, such as applog and PCAP logs, by running the failed test cases in a sequence.

Figure 2-23 Logging/Rerun new stage



If the `Fetch_Log_Upon_Failure` parameter is set to YES and if any test case fails in the initial run, then:

- The failed test case reruns and log collection start in the Logging/Rerun stage after the initial run is completed for all the test cases.
- The logs from the initial execution are collected, but they might be incorrect.
- Even if the `rerun` parameter is set to 0, the failed test case reruns in the Logging/Rerun stage and the log is collected.

> **Note**
>
> Not applicable for all the NFs.

- If the `Fetch_Log_Upon_Failure` parameter is set to NO and if any test case fails in the initial run, then the failed test case rerun starts in the same stage after the initial execution is over for all the test cases in its group.

2.8 ATS Maintenance Scripts

ATS maintenance scripts are used to perform the following operations:

- Taking a backup of the ATS custom folders and Jenkins pipeline.
- Viewing the configuration and restoring the Jenkins pipeline.
- Viewing the configuration and installing or uninstalling ATS and stubs.

ATS maintenance scripts are present in the ATS image at the following path: `/var/lib/jenkins/ocats_maint_scripts`

Run the following command to copy the scripts to a local system (bastion):

```
kubectl cp <NAMESPACE>/<POD_NAME>:/var/lib/jenkins/ocats_maint_scripts
<DESTINATION_PATH_ON_BASTION> pod
```

For example,

```
kubectl cp ocpcf/ocats-ocats-policy-694c589664-js267:/var/lib/Jenkins/
ocats_maint_scripts /home/meta-user/ocats_maint_scripts pod
```

2.8.1 ATS Scripts

ATS maintenance scripts are used to perform various task related to ATS and Jenkin pipeline.

The following are the types of scripts:

- **ats_backup.sh**: This script requires the user's input and takes a backup of the ATS custom folders, Jenkins jobs, and user's folders on the user's system. The backup can be of the Jenkins jobs and user's folder, the custom folders, or both. The custom folders include cust_regression, cust_newfeatures, cust_performance, cust_data, and custom_config. For a Jenkins job or a user's folder, the script only takes a backup of the config.xml file. Also, the script requires the user to store a backup on the user's system (the default path is the location from where the script is being run) and to create a backup folder on the system and take the backup of the chosen folder from the corresponding ATS into the backup folder. The backup folder name can be of the following notation: `ats_<version>_backup_<date>_<time>`.
- **ats_uninstall.sh**: This script requires the user's input and uninstalls the corresponding ATS.
- **ats_install.sh**: This script requires the user's input and installs a new ATS. If `PVEnabled` is set to `true`, the script also reads the PVC name from values.yaml and creates values.yaml before installation. Also, if needed, the script performs the postinstallation steps, such as copying tests and Jenkins jobs' folders from the `ats_data` tar file to the ATS pod when PV is deployed, and then restarts the pod.
- **ats_restore.sh**: This script requires the user's inputs, restores the new release ATS pipeline, and views the configuration by referring to the last release ATS Jenkins jobs and the user's configuration. It depends on the user whether to use the backup folders from the user's system to restore the ATS configuration. If the user instructs the script to use the backup from the system, the script requires the path of the backup and uses the backup to restore. Otherwise, the script requires the last ATS Helm release name to refer to its Jenkins jobs and the user's configuration to restore.
  The script refers to the last release of ATS Jenkins pipelines and sets the `Discard old builds` property if this property is set in the last release of ATS for a pipeline but not in the current release. If this property is set in both releases, the script just updates the values according to the last release. Also, the script restores the `pipeline environment variables` values as per the last release of ATS. If any custom pipeline (created by the user) was present in the last release of ATS, the script restores that as well. It also restores the extra views created by NF users, for example, policy users, SCP users, and NRF users. Moreover, the script displays messages about the pending configuration that the user needs to perform manually. For example, a new pipeline or a new environment variable (for a pipeline) is introduced in the new release.

  While deploying ATS without PV, Jenkins needs to be restarted for the restore process to complete. If the last release ATS contains the `Configuration_Type` parameter, the `Configuration_Type` script needs to be approved with the **In Process Script Approval** setting under **Manage ATS** in Jenkins for the restore process to complete.

2.8.2 Updating ats_install.sh

Currently, the `ats_install.sh` script copies the tests folder and Jenkins jobs folder into the ATS pod and then restarts the pod when deployed with PV.

How to Update ats_install.sh

Other NFs can also use the `ats_install.sh` scripts. However, additional post installation steps may have to be performed manually for a few NFs.

For the additional post installation commands, perform the following steps:

1. In the `ats_install.sh` script, there is a post install section between `####POST_INSTALL_START####` and `#### POST_INSTALL_END ####`.
   a. Add the required post install commands.

   > **Note**
   >
   > These commands are NF-specific.

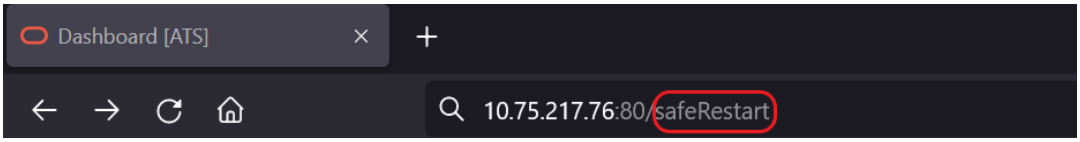
   b. Use the following commands:
      - `$namespace` for the namespace value
      - `$pod_name` for the pod name
      - `$ats_data_path` for the `ats_data` folder path (it has tests folder and Jenkins jobs folder provided as tar file in ATS package)
   c. In the if-else block related to whether PV is enabled or not, add the following:
      - Add a command specific to `PVEnabled=true` in the `if block`.
      - Add a command specific to `PVEnabled=false` in the `else block`.
2. For additional inputs, enter the required code between `#### INPUT_START ####` and `#### INPUT_END ####`.

2.8.3 Restarting Jenkins without Restarting Pod

Perform the following procedure to restart Jenkins without restarting pods:

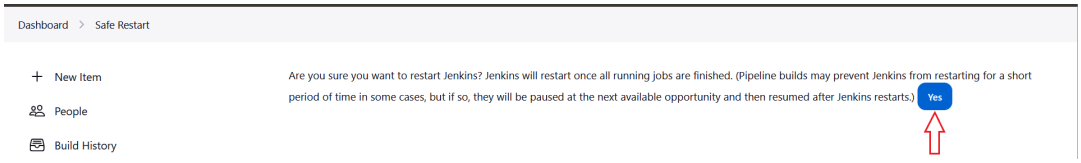
1. Log in as the Jenkins admin.
2. Go to the `<Jenkins_IP>:<port>/safeRestart`, for example, `10.87.73.32:32156/safeRestart`.

Figure 2-24 Safe Restart



3. Click **Yes**.

Figure 2-25 Restart Jenkins



2.8.4 Updating Stub Scripts

The following stubs can be updated:

- **stub_uninstall.sh**: This script requires the user's inputs and uninstalls all the stubs.
- **stub_install.sh**: This script requires the user's inputs and installs all the stubs.

> **Note**
>
> Currently, `stub_uninstall.sh` and `stub_install.sh` work.

Perform the following procedure to update the stub scripts for other NFs (NRF in this case):

1. Go to the `stub` folder.
2. From each script:
   a. Remove the CNC Policy-specific stubs inputs (dns, amf, and ldap), and add the input code blocks for NRF-specific stubs.
   b. For the stubs to uninstall, change the value of the `stubUninstallList` variable, and delete the variables for the CNC Policy-specific stubs below it.

      > **Note**
      >
      > `stubUninstallList` contains the Helm release names of the common stubs that are deployed generally.

   c. Declare the variables for the NRF-specific stubs below the `stubUninstallList` line.
   d. Remove the Helm uninstallation commands of the policy-specific stubs, and add the Helm uninstallation commands of the NRF-specific stubs.
   e. For the stubs to install, change the value of the `stubInstallList` variable, and delete the variables for the CNC Policy-specific stubs below it.

      > **Note**
      >
      > `stubInstallList` contains the Helm release names of the common stubs that are deployed generally.

   f. Declare the variables for the NRF-specific stubs below the `stubInstallList` line.
   g. Remove the Helm installation commands of the CNC Policy-specific stubs, and add the Helm installation commands of the NRF-specific stubs.

2.8.5 Running ATS and Stub Deployment Scripts

Perform the following procedure to run ATS and stub deployment scripts:

> **Note**
>
> If you want to take a backup of the custom folders or Jenkins jobs and user's configuration or both, run the `ats_backup.sh` script.

1. Run the `ats_install.sh` script to install the new release ATS (values.yaml of the ATS Helm chart must be updated before this step).
2. Run the `ats_restore.sh` script to restore the new ATS pipeline and view configuration.

   > **Note**
   >
   - You might perform the manual steps required for the restore script.
   - You must copy all the necessary changes to the new release ATS from the last release ATS. To get the changes in the last release, you must refer to the custom folders in the last release ATS backup on the system with an existing backup using `ats_backup.sh` before this step.
   - You can remove the last release ATS pod using the `ats_uninstall.sh` script while continuing to retain the last release PVC. You can use the last release PVC to port backward. Delete the last release PVC when you do not require the backward porting.

3. Run the `stub_install.sh` script to install all the new release stubs values.yaml of the stub Helm charts must be updated before this step.
4. Run the `stub_uninstall.sh` script to uninstall all the last release stubs.

2.9 ATS System Name and Version Display on the ATS GUI

This feature displays the ATS system name and version on the ATS GUI.

You can log in to the ATS application using the login credentials to view the following:

- ATS system name: Abbreviated product name followed by NF name.
- ATS Version: Release version of ATS.

2.10 ATS Tagging Support

The ATS Tagging Support feature assists in running the feature files after filtering features and scenarios based on tags. Instead of manually navigating through several feature files, the user can save time by using this feature.

The GUI offers the following four options for selecting tag types:

- Feature_Include_Tags: The features that contain either of the tags available in the **Feature_Include_Tags** field are considered for tagging.

  - For example, "cne-common", "config-server". All the features that have either "cne-common" or "config-server" tags are taken into consideration.

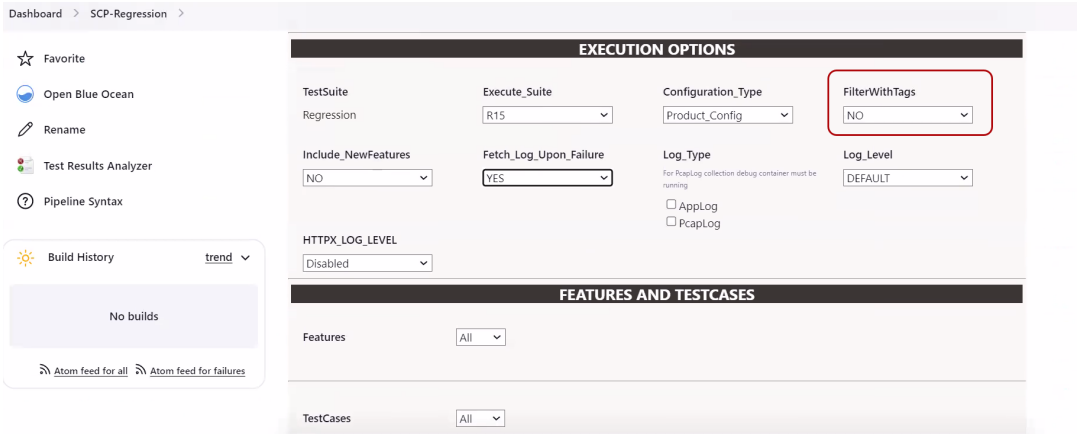
- Feature_Exclude_Tags: The features that contain neither of the tags available in the **Feature_Exclude_Tags** field are considered for tagging.
  - For example, "cne-common","config-server". All the features that have neither "cne-common" nor "config-server" as tags are taken into consideration.
- Scenario_Include_Tags: The scenarios that contain either of the tags available in the **Scenario_Include_Tags** field are considered.
  - For example, "sanity", "cleanup". The scenarios that have either "sanity" or "cleanup" tags are taken into consideration.
- Scenario_Exclude_Tags: The features that contain neither of the tags available in the **Scenario_Exclude_Tags** field are considered.
  - For example, "sanity", "cleanup". The scenarios that have neither "sanity" nor "cleanup" as tags are taken into consideration.

Filter with Tags

The procedure to filter feature files and scenarios based on tags are as follows:

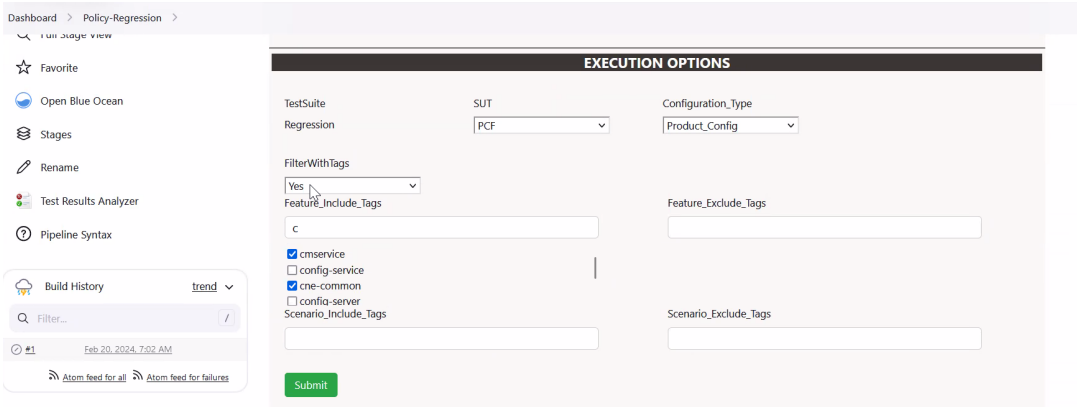
1. On the NF home page, click any new feature or regression pipeline, where you want to use this feature.
2. In the left navigation pane, click **Build with Parameters**. The following image appears.

Figure 2-26 Filter with Tags



3. Select **Yes** from the **FilterWithTags** drop-down menu. The result shows four input fields. The default value of **FilterWithTags** field is "No".
4. The input fields serve as a search or filter, displaying all tags that match the prefix entered. You can select one or multiple tags.

Figure 2-27 Tags Matching with Entered Prefix



5. Select the required tags from the different tags list and click **Submit**.

The specified feature-level tags are used to filter out features that contain any one of the include tags and none of the exclude tags. Here, any or both the fields may be left empty. All features are automatically taken into consideration when both fields are empty.

The scenario level tags are used to filter out the scenarios from the features filtered above. Only scenarios with any of the include tags and none of the exclude tags are considered. Any or both fields can be empty. When both fields are empty, all the scenarios from the above filtered feature files are considered.

> **Note**
>
> - If you select the Select_Option as 'All', all the displayed features and scenarios will run.
> - If you select the Select_Option as 'Single/MultipleFeatures, it enables you to select some features, and only those features and respective scenarios are going to run.

2.10.1 Combination of Tags and their Results

The combination of tags and expected results are as follows.

Table 2-6 Result of Filtered Tags

Feature_Include	Feature_Exclude	Scenario_Include	Scenario_Exclude	Results
-	-	-	-	All the features and scenarios are taken into consideration.
"abc","def"	-	-	-	Features with either "abc" or "def" tags and all scenarios from the filtered features are taken into consideration.

Table 2-6 (Cont.) Result of Filtered Tags

Feature_Include	Feature_Exclude	Scenario_Include	Scenario_Exclude	Results
-	"abc","def"	-	-	All the features with neither "abc" nor "def" tags and all scenarios from the filtered features are taken into consideration.
-	-	"sanity","cne"	-	Scenarios with either "sanity" or "cne" tags and features having these scenarios are taken into consideration.
-	-	-	"sanity","cne"	Scenarios with neither "sanity" nor "cne" tags and features having these filtered scenarios are taken into consideration.
"abc","def"	"ghi"	-	-	Features with either "abc" or "def" tags but without the "ghi" tag and all scenarios from filtered features are taken into consideration.
"abc","def"	-	"sanity","cne"	-	Scenarios only with either "sanity" or "cne" tags and only features that contain these scenarios and have either "abc" or "def" as feature tags are taken into consideration.
"abc","def"	-	-	"sanity","cne"	Scenarios with neither "sanity" nor "cne" tags and only features that contain the filtered scenarios and have either "abc" or "def" feature tags are taken into consideration.
-	"ghi"	"sanity","cne"	-	Features without the "ghi" tag and scenarios with either "sanity" or "cne" tags from the filtered features are taken into consideration.
-	"ghi"	-	"sanity","cne"	Features without the "ghi" tag and scenarios without the "sanity" and "cne" tags from filtered features are taken into consideration.

Table 2-6 (Cont.) Result of Filtered Tags

Feature_Include	Feature_Exclude	Scenario_Include	Scenario_Exclude	Results
-	-	"sanity","cne"	"cleanup"	Scenarios with either the "sanity" or "cne" tags and without the "cleanup" tag and features with filtered scenarios are taken into consideration.
"abc","def"	"ghi"	"sanity","cne"	-	Scenarios with either the "sanity" or "cne" tags and features that have these scenarios and have either the "abc" or "def" tags but not the "ghi" tag are taken into consideration.
"abc","def"	-	"sanity","cne"	"cleanup"	Scenarios with either the "sanity" or "cne" tags and without the "cleanup" tag, and features having the filtered scenarios and having the feature tags either "abc" or "def" are taken into consideration.
"abc","def"	"ghi"	-	"cleanup"	Scenarios without the tag "cleanup", and features with filtered scenarios and having either "abc" or "def" as feature tags but not the "ghi" tag are taken into consideration.
-	"ghi"	"sanity","cne"	"cleanup"	Scenarios with either the "sanity" or "cne" tags and without the "cleanup" tag, and features with filtered scenarios and not the tag "ghi," are taken into consideration.
"abc","def"	"ghi"	"sanity","cne"	"cleanup"	Scenarios with either "sanity" or "cne" tags and without the "cleanup" tag, and features with filtered scenarios and feature tags either "abc" or "def" but without the tag "ghi" are taken into consideration.

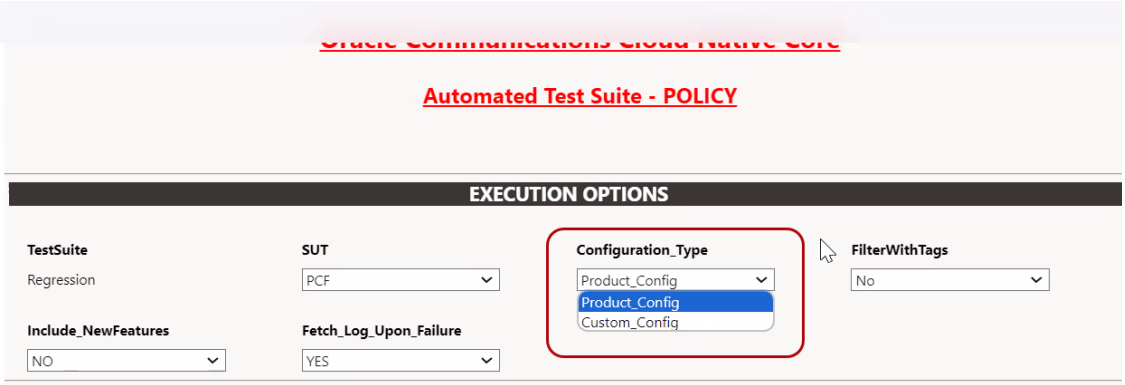
> **Note**
>
> - The tags mentioned in the table are just examples; they may or may not be actually used.
> - The **Replay** option in the Jenkins GUI is not supported for tag-related test case execution. Always trigger builds related to tagging from the **Build with Parameter** step, and do not replay any previous builds.
> -

2.11 Custom Folder Implementation

The Custom Folder Implementation feature allows the user to update, add, or delete test cases without affecting the original product test cases in the new features, regression, and performance folders. The implemented custom folders are cust_newfeatures, cust_regression, and cust_performance. The custom folders contain the newly created, customised test cases.

Initially, the product test case folders and custom test case folders will have the same set of test cases. The user can perform customization in the custom test case folders, and ATS always runs the test cases from the custom test case folders. If the option "Configuration_Type" is present on the GUI,the user needs to set its value to "Custom_Config" to populate test cases from the custom test case folders.

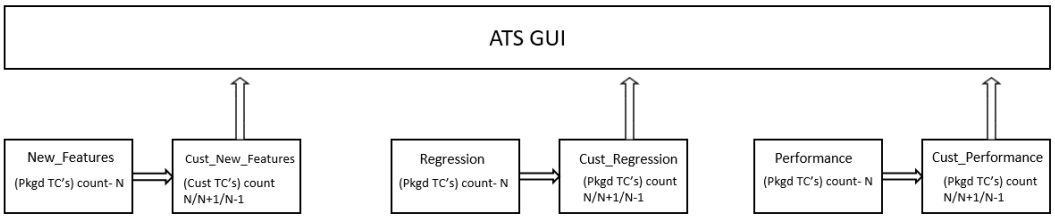
Figure 2-28 Custom Config Folder



Summary of Custom Folder Implementation

- Separate folders such as cust_newfeatures, cust_regression, and cust_performance are created to hold the custom cases.
- The prepackaged test cases are available in the newfeature and regression Folder.
- The user copies the required test cases to the cust_newfeatures and cust_regression folders, respectively.
- Jenkins always points to the cust_newfeatures and cust_regression folders to populate them in the menu.
  If someone initially launches ATS, they will not see any test cases in the menu if the cust folders are not populated. To avoid this, it is recommended to prepopulate both the folders, cust and original, and ask the user to modify only the cust folder if needed.

Figure 2-29 Summary of Custom Folder Implementation



2.12 Health Check

Health Check functionality is to check the health of the System Under Test (SUT)

Earlier, ATS used Helm test functionality to check the health of the System Under Test (SUT). With the implementation of the ATS Health Check pipeline, the SUT health check process has been automated. ATS health checks can be performed on webscale and non-webscale environments.

Convert a Value in Base64

The following command can be utilized to convert any value into base64 encoding:

```
echo-n "value"| base64
```

For example,

```
echo-n "126.98.76.43"| base64
```

Deploying Health Check in a Webscale Environment

- Set the `Webscale` to 'true' and the following parameters by encoding them with base64 in the ATS values.yaml file:
- Set the following parameter to encrypted data:

```
webscalejumpserverip: encrypted-data
webscalejumpserverusername: encrypted-data
webscalejumpserverpassword: encrypted-data
webscaleprojectname: encrypted-data
webscalelabserverFQDN: encrypted-data
webscalelabserverport: encrypted-data
webscalelabserverusername: encrypted-data
webscalelabserverpassword: encrypted-data
```

Encrypted data is the value of parameters encrypted in base64. Fundamentally, Base64 is used to encode the parameters.

For example:

```
webscalejumpserverip=$(echo -n '10.75.217.42' | base64), Where Webscale Jump
server ip needs to be provided
webscalejumpserverusername=$(echo -n 'cloud-user' | base64), Where Webscale
```

```
Jump server Username needs to be provided
webscalejumpserverpassword=$(echo -n '*****' | base64), Where Webscale Jump
server Password needs to be provided
webscaleprojectname=$(echo -n '*****' | base64), Where Webscale Project Name
needs to be provided
webscalelabserverFQDN=$(echo -n '*****' | base64), Where Webscale Lab Server
FQDN needs to be provided
webscalelabserverport=$(echo -n '*****' | base64), Where Webscale Lab Server
Portneeds to be provided
webscalelabserverusername=$(echo -n '*****' | base64), Where Webscale Lab
Server Username needs to be provided
webscalelabserverpassword=$(echo -n '*****' | base64), Where Webscale Lab
Server Password needs to be provided
```

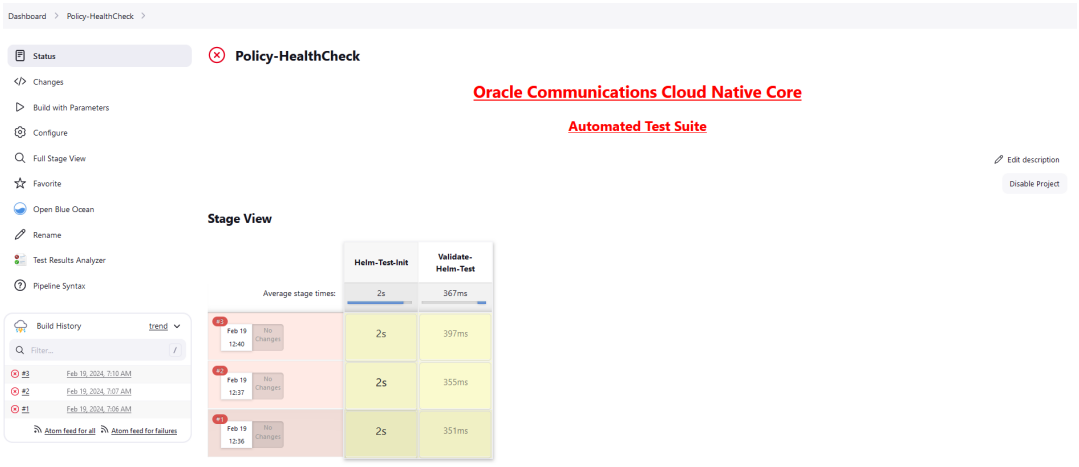
Running Health Check Pipeline in an Webscale Environment

To run Health Check pipeline:

1. Log in to ATS using respective <NF> login credentials.
2. Click <NF>HealthCheck pipeline and then click **Configure**.

> **Note**
>
> <NF> denotes the network function. For example, in Policy, it is called as Policy-HealthCheck pipeline.

Figure 2-30 Configure Healthcheck



3. Provide parameter a with Helm release name deployed. If there are multiple releases, use comma to provide all Helm release names.

   ```
   //a = helm releases [Provide Release Name with Comma Separated if more
   than 1 ]
   ```

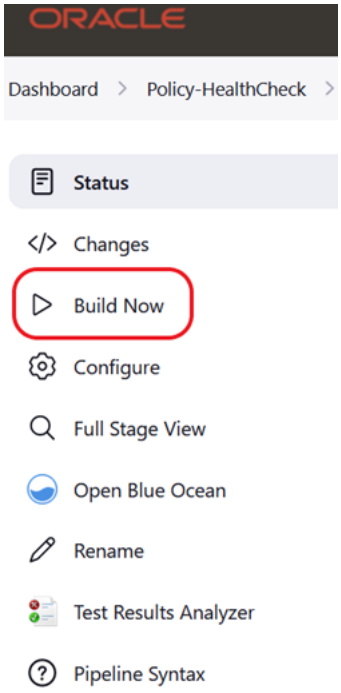
   Provide parameter c with the appropriate Helm command, such as helm, helm3, or helm2.

   ```
   //c = helm command name [helm or helm2 or helm3]
   ```

Figure 2-31 Save the Changes

4. Save the changes and click **Build Now**. ATS runs the health check on respective network function.

Figure 2-32 Build Now



Deploying Health Check Pipeline in an OCI Environment

To use a ssh private key, create healthcheck-oci-secret and set the value of the key "passwordAuthenticationEnabled" to false.

Creating healthcheck-oci-secret

Create healthcheck-oci-secret to use ssh private keys instead of passwords using the following command:

```
kubectl create secret generic healthcheck-oci-secret --from-
file=bastion_key_file='<path of bastion ssh private key file>' --from-
file=operator_instance_key_file='<path of operator instance ssh private key
file>' -n <ATS namespace>
```

For example,

```
kubectl create secret generic healthcheck-oci-secret --from-
file=bastion_key_file='/tmp/bastion_private_key' --from-
file=operator_instance_key_file='/tmp/operator_instance_private_key' -n
seppsvc
```

> **Note**
>
> - Maintain the name of the secret as "healthcheck-oci-secret".
> - Ensure that the '--from-file' keys retain the same names: "bastion_key_file" and "operator_instance_key_file".
> - If the SSH private key is identical for both the bastion and operator instance, you can use the same path for both in the secret creation command.

Perform the following procedure to deploy ATS Health Check in a OCI environment:

Set the `Webscale` parameter set to 'false' and following parameters by encoding it with base64 in the ATS values.yaml file.

- To use password, provide base64 encoded values for key "password" for both bastion and operator instances, and set the value of key **passwordAuthenticationEnabled** to "true".
- Set the following parameter to encrypted data:

```
envtype: encrypted-data
ociHealthCheck:
  passwordAuthenticationEnabled: true or false
  bastion:
    ip: encrypted-data
    username: encrypted-data
    password: encrypted-data
  operatorInstance:
    ip: encrypted-data
    username: encrypted-data
    password: encrypted-data
```

> **Note**
>
> All fields are mandatory except for passwords. When the "passwordAuthenticationEnabled" field is set to true, only the "password" field needs to be updated; otherwise, it can remain with its default value.

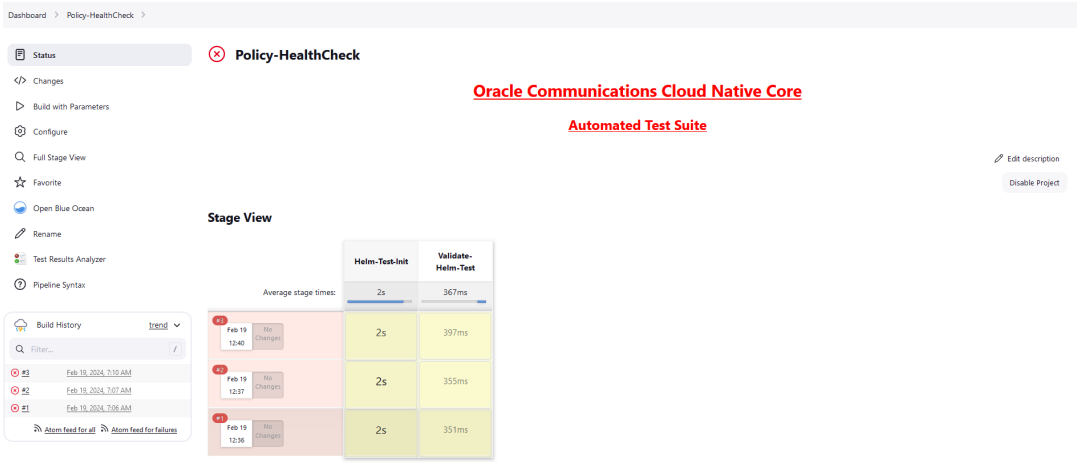
Running Health Check Pipeline in an OCI Environment

To run ATS Health Check pipeline:

1. Log in to ATS using respective <NF> login credentials.
2. Click <NF>HealthCheck pipeline and then click **Configure**.

> **Note**
>
> <NF> denotes the network function. For example, in Policy, it is called as Policy-HealthCheck pipeline.

Figure 2-33 Configure Healthcheck



3. Provide parameter a with Helm release name deployed. If there are multiple releases, use comma to provide all Helm release names.

```
//a = helm releases [Provide Release Name with Comma Separated if more
than 1 ]
```

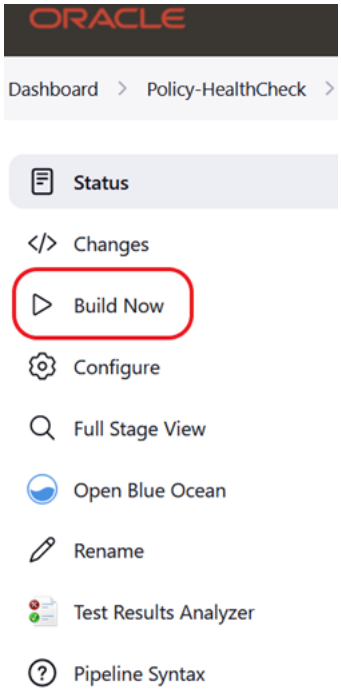
Provide parameter c with the appropriate Helm command, such as helm, helm3, or helm2.

```
//c = helm command name [helm or helm2 or helm3]
```

Figure 2-34 Save the Changes

4. Save the changes and click **Build Now**. ATS runs the health check on respective network function.

Figure 2-35 Build Now



Deploying Health Check in a Non-Webscale or Non-OCI Environment

Perform the following procedure to deploy ATS Health Check in a non-webscale or non-OCI environment such as OCCNE:

Set the `Webscale` parameter set to 'false' and following parameters by encoding it with base64 in the ATS values.yaml file:

```
occnehostip: encrypted-data
occnehostusername: encrypted-data
occnehostpassword: encrypted-data
```

Example:

```
occnehostip=$(echo -n '10.75.217.42' | base64) , Where occne host ip needs to
be provided
occnehostusername=$(echo -n 'cloud-user' | base64), Where occne host username
needs to be provided
occnehostpassword=$(echo -n '****' | base64), Where password of host needs to
be provided
```

Running Health Check Pipeline in a Non-Webscale or Non-OCI Environment

Perform the following procedure to run the ATS Health Check pipeline in a non-webscale or non-OCI environment such as OCCNE:

1. Log in to ATS using respective <NF> login credentials.
2. Click <NF>HealthCheck pipeline and then click **Configure**.
3. Provide parameter a with Helm release name deployed. If there are multiple releases, use comma to provide all Helm release names.

Provide parameter b with SUT deployed namespace name.

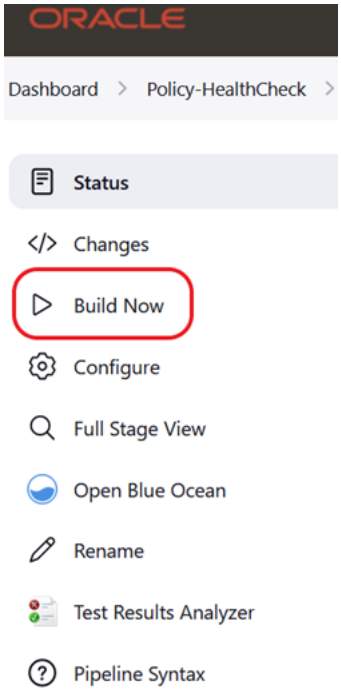
Provide parameter c with the appropriate Helm command, such as helm, helm3, or helm2.

```
//a = helm releases [Provide Release Name with Comma Separated if more
than 1 ]
//b = Namespace, If not applicable to WEBSCALE environment then remove the
argument
//c = helm command name [helm or helm2 or helm3]
```

Figure 2-36 Save the Changes

4. Save the changes and click **Build Now**. ATS runs the health check on respective network function.

Figure 2-37 Build Now



By clicking **Build Now**, you can run the health check on ATS and store the result in the console logs.

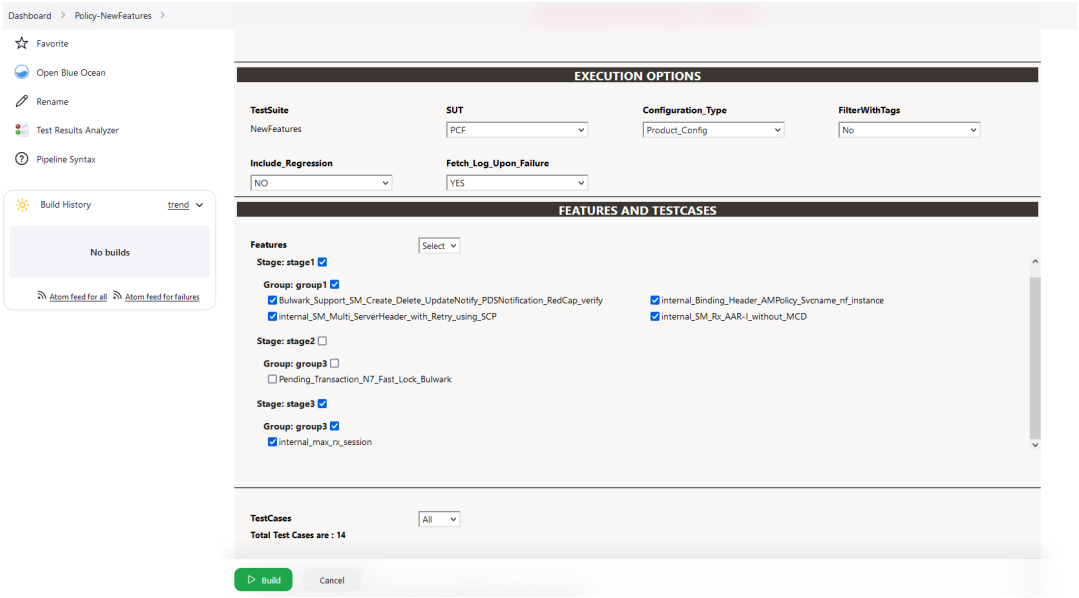
2.13 Individual Stage Group Selection

The Individual Stage Group Selection feature allows you to select and execute a single or multiple stages or groups by selecting a check box for the corresponding stage or group.

Follow the steps to select an individual stage or group:

1. Click NF-Regression or NF-NewFeatures, and then click **Build with Parameters**.
2. On the FEATURE and TESTCASES section, click **Select** from the **Features** drop-down menu.
3. Select the corresponding check box to select any number of stages or groups you want to run from the list available for execution.

Figure 2-38 Stages or Groups Selection

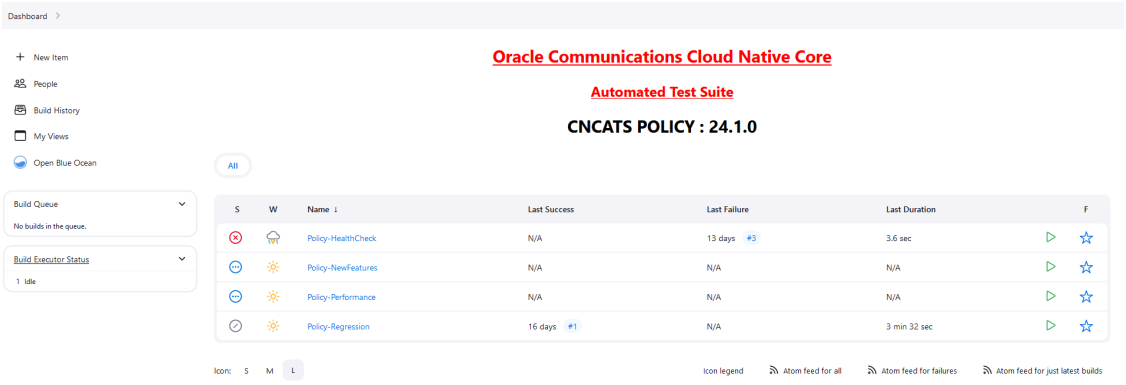


4. Scroll down to click **Build**.

2.14 Lightweight Performance

The Lightweight Performance feature allows you to run performance test cases. In ATS, a new pipeline known as "<NF>-Performance", where NF stands for Network Function, is introduced, for example, Policy-Performance.

Figure 2-39 Sample Screen: Home Page



The <NF>-Performance pipeline verifies from 500 to 1k TPS (Transactions per Second) of traffic using the http-go tool, a tool used to run the traffic on the backend. It also helps to monitor the CPU and memory of microservices while running lightweight traffic.

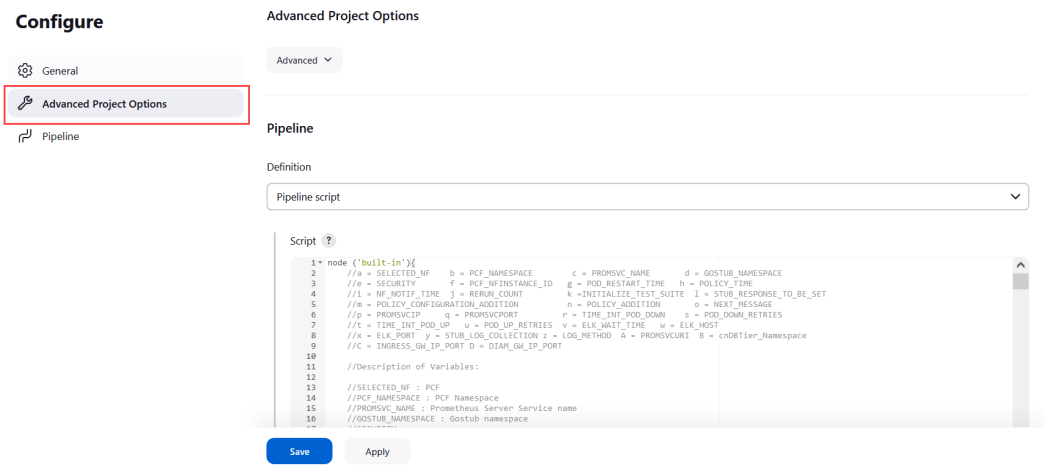
The duration of the traffic run can be configured on the pipeline.

2.14.1 Configure <NF>-Performance Pipeline

Perform the following to configure the performance pipeline:

1. On the NF home page, click **<NF>-Performance** pipeline, and then click **Configure**. The **General** tab appears. The user must wait for the page to load completely.
2. Click the **Advanced Project Options** tab. Scroll down to reach the **Pipeline** configuration section.

Figure 2-40 Advanced Project Options



3. Update the configurations as per your NF requirements and click **Save**. The **Pipeline <NF>-Performance** page appears.
4. Click **Build Now**. This triggers lightweight traffic for the respective network function.

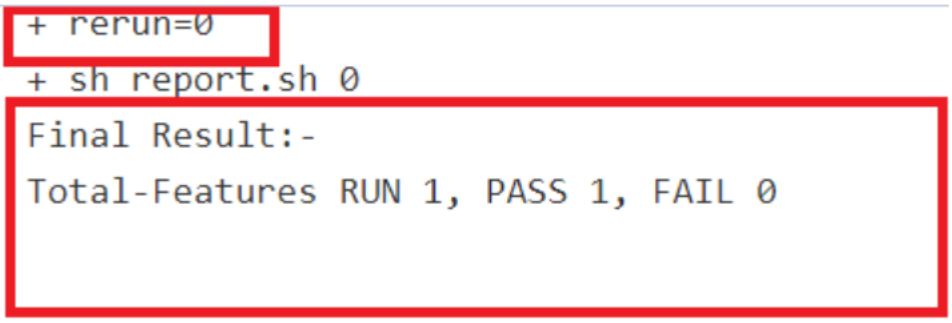
2.15 Managing Final Summary Report, Build Color, and Application Log

This feature displays an overall execution summary, such as the total run count, pass count, and fail count.

Supports Implementation of Total-Features
ATS supports implementation of **Total-Features** in the final summary report. Based on the rerun value set, the **Final Result** section in the final summary report displays the Total-Features output.

- If rerun is set to 0, the test result report shows the following result:

Figure 2-41 Total-Features = 1, and Rerun = 0



- If rerun is set to non-zero, the test result report shows the following result:

Figure 2-42 Total-Features = 1, and Rerun = 2

```
+ rerun=2
+ sh report.sh 2
Initial Run :-
Features RUN 1, PASS 0, FAIL 1

1st Rerun:-
Features RUN 1, PASS 0, FAIL 1

2nd Rerun:-
Features RUN 1, PASS 0, FAIL 1

Final Result:-
Total-Features RUN 1, PASS 0, FAIL 1
```

Changes After Parallel Test Execution Framework Feature Integration
After incorporating the Parallel Test Execution feature, the following results were obtained:

Final Summary Report Implementations

Figure 2-43 Group Wise Results

```
**********Stage: stage1 Group: group3 Result**********
Initial Run :-
Features RUN 1, PASS 0, FAIL 1

1st Rerun:-
Features RUN 1, PASS 0, FAIL 1

2nd Rerun:-
Features RUN 1, PASS 0, FAIL 1

stage1 group3
Final Result:-
Total-Features RUN 1, PASS 0, FAIL 1
```

Figure 2-44 Overall Result When Selected Feature Tests Pass

```
**********Overall Result**********

Took 0h0m50s

Final Result:-
Total-Features RUN 1, PASS 1, FAIL 0
```

Figure 2-45 Overall Result When Any of the Selected Feature Tests Fail

```
**********************************Overall Result****************************************
Failing scenarios:
Stage:stage1 Group:group3
  SCP_Registration_With_PLMNList.feature:45  Static configuration to NRF for all nfType

Took 0h1m59s

Initial Run:-
Features RUN 1, PASS 0, FAIL 1

1st Rerun:-
Featues RUN 1, PASS 0, FAIL 1

2nd Rerun:-
Featues RUN 1, PASS 0, FAIL 1

Final Result:-
Total-Featues RUN 1, PASS 0, FAIL 1
```

Display the Count and Names of Skipped Features Whenever Skipped During a Rerun
Starting from the 25.2.200 release, ATS includes additional details in the Overall Result Summary by showing the count and names of any features that are skipped during a rerun. The following screenshot depicts this scenario:

Figure 2-46 Overall Result Summary Whenever Features Skipped During a Rerun

```
+ set +x
**********************************Overall Result****************************************
Failing scenarios:
Stage:stage1 Group:group1
  YamlSchema_Import_Export/YamlSchema_Import_Export.feature:58  Re_Import_YamlSchema_Verify_SMPolicy
Stage:stage2 Group:group3
  N7_KPIs/N7_KPIs.feature:55  N7_KPIs_Create_Update_Delete
Stage:stage3 Group:group1
  Dynamic_Policy_Override/Dynamic_Policy_Override.feature:25  Dynamic_Override_Qos_Data

Took 0h12m45s

Initial Run:-
Features RUN 3, PASS 0, FAIL 3

1st Rerun:-
Features RUN 2, PASS 0, FAIL 2

2nd Rerun:-
Features RUN 2, PASS 0, FAIL 2

Final Result:-
Total-Features RUN 3, PASS 0, FAIL 3


Total skipped features count: 1
stage: stage3  Group: group1
  Dynamic_Policy_Override/Dynamic_Policy_Override.feature:25
[Pipeline] }
[Pipeline] // script
[Pipeline] }
```

Implementing Build Colors

ATS supports implementation of build color. The details are as follows:

Table 2-7 Build Color Details

Rerun Values	Rerun set to zero		Rerun set to non-zero		
Status of Run	**All Passed in Initial Run**	**Some Failed in Initial Run**	**All Passed in Initial Run**	**Some Passed in Initial Run, Rest Passed in Rerun**	**Some Passed in Initial Run, Some Failed Even After Rerun**
Build Status	SUCCESS	FAILURE	SUCCESS	SUCCESS	FAILURE
Pipeline Color	GREEN	Execution Stage where test cases failed shows YELLOW color, rest of the successful stages are GREEN.	GREEN	GREEN	Execution Stage where test cases failed shows YELLOW color, rest of the successful stages are GREEN
Status Color	BLUE	RED	BLUE	BLUE	RED

Changes After Integrating Parallel Test Execution Framework Feature
In sequential execution, the build color or overall pipeline status of any run was mainly dependent on the following parameters:

- the rerun count and the pass or fail status of test cases in the initial run
- the rerun count and the pass or fail status of test cases in the final run

For the parallel test case execution, the pipeline status also depends on another parameter, "Fetch_Log_Upon_Failure," which is given in the **build with parameters** page. If the parameter `Fetch_Log_Upon_Failure` is not there, its default value is considered "NO".

Table 2-8 Pipeline Status When Fetch_Log_Upon_Failure = NO

Rerun Values	Rerun set to zero		Rerun set to non-zero		
Passed/Failed	**All Passed in Initial Run**	**Some Failed in Initial Run**	**All Passed in Initial Run**	**Some Passed in Initial Run, Rest Passed in Rerun**	**Some Passed in Initial Run, Some Failed Even After Rerun**
Status	SUCCESS	FAILURE	SUCCESS	SUCCESS	FAILURE

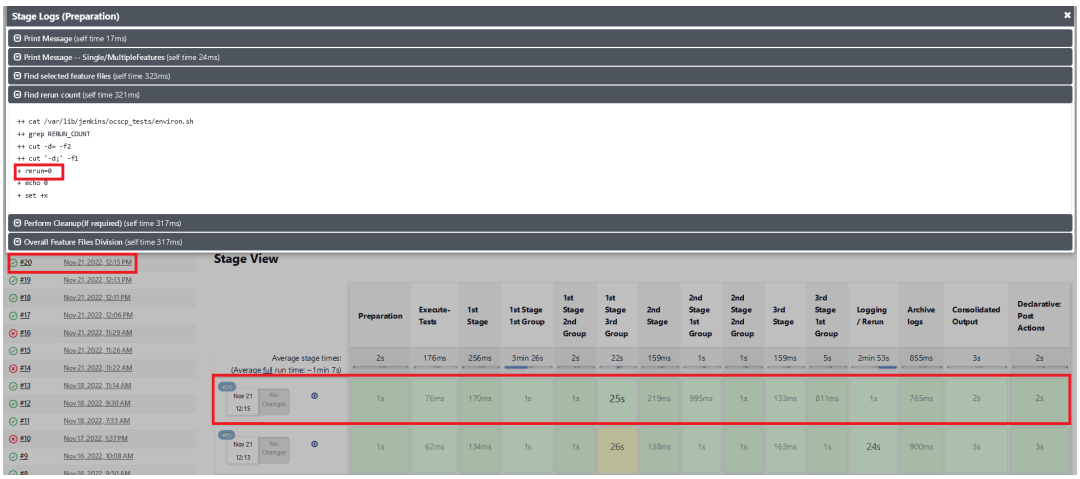
Table 2-9 Pipeline Status When Fetch_Log_Upon_Failure = YES

Rerun Values	Rerun set to zero			Rerun set to non-zero		
Passed/Failed	**All Passed in Initial Run**	**Some Failed in Initial Run and Failed in Rerun**	**Some Failed in Initial Run and Passed in Rerun**	**All Passed in Initial Run**	**Some Passed in Initial Run, Rest Passed in Rerun**	**Some Passed in Initial Run, Some Failed Even After Rerun**
Status	SUCCESS	FAILURE	SUCCESS	SUCCESS	SUCCESS	FAILURE

Some common combinations of these parameters, such as `rerun_count`, `Fetch_Log_Upon_Failure`, and `pass/fail status of test cases in initial and final run` and the corresponding build colors are as follows:

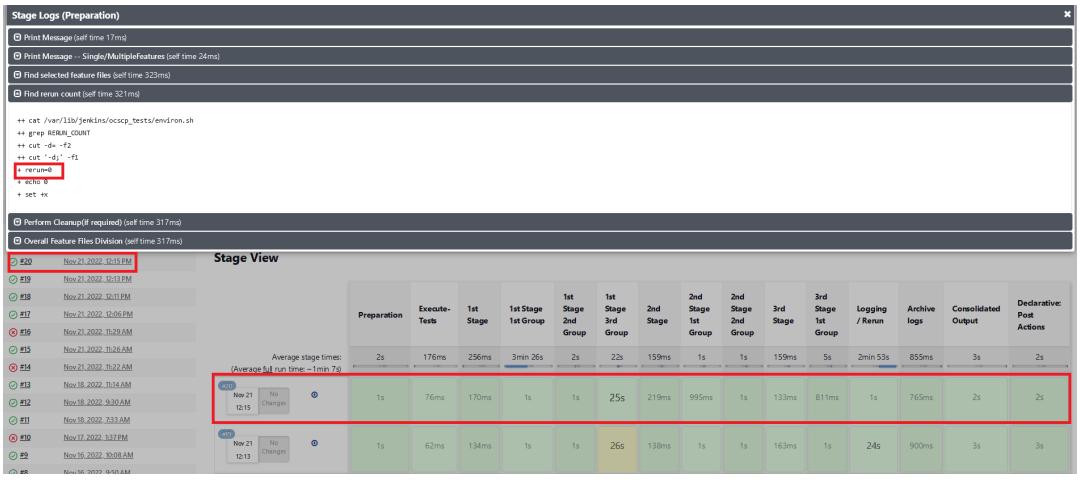
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases pass in the initial run. The pipeline will be green, and its status will show as blue.

Figure 2-47 Fetch_Log_Upon_Failure is set to YES and rerun_count is set to 0, test cases pass



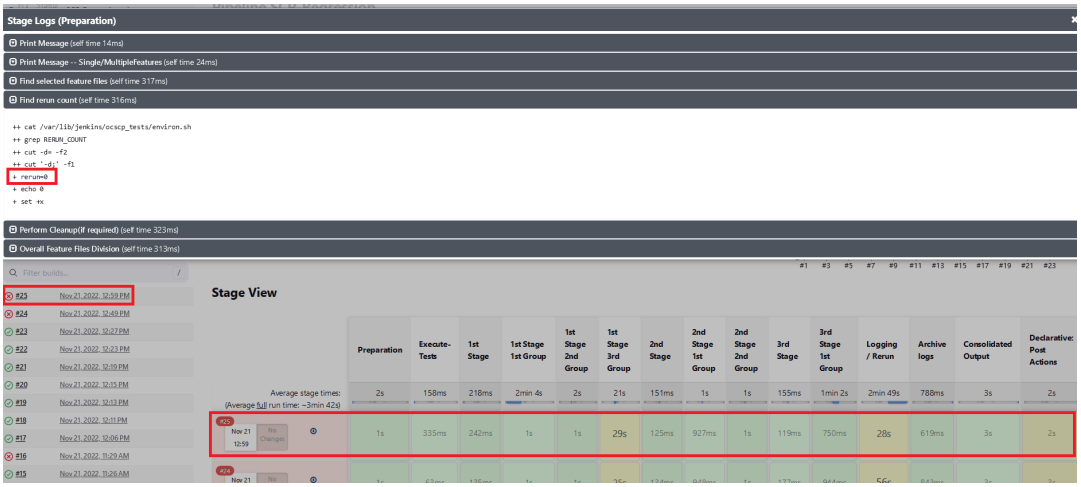
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases fail on the initial run but pass during the rerun. The initial execution stage is yellow and all subsequent successful stages will be green, and the status will be blue.

Figure 2-48 Test Cases Fail on the Initial Run but Pass in the Rerun



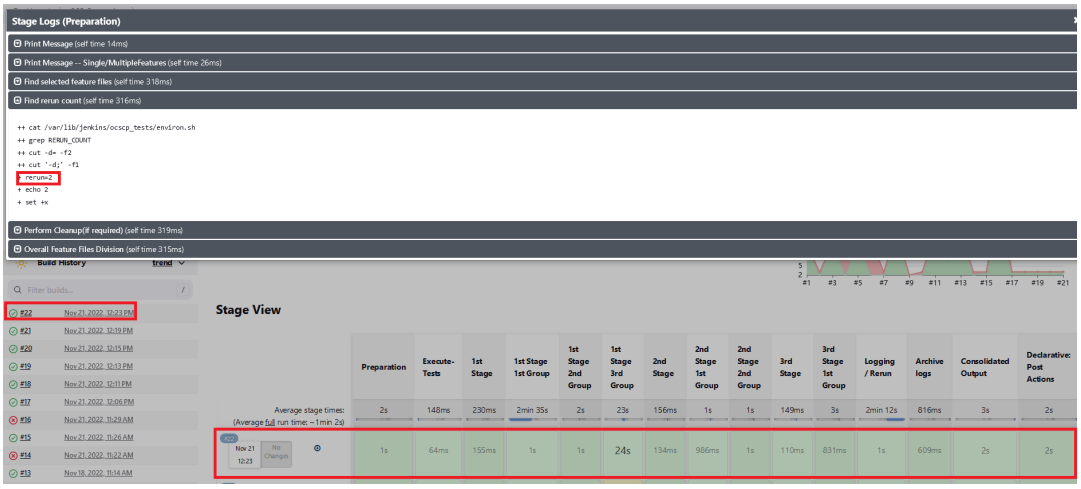
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases fail in both the initial and the rerun. Execution stages will show as yellow, all other successful stages will be shown as green, and the overall pipeline status will be red.

Figure 2-49 Test Cases Fail in Both the initial and the Rerun



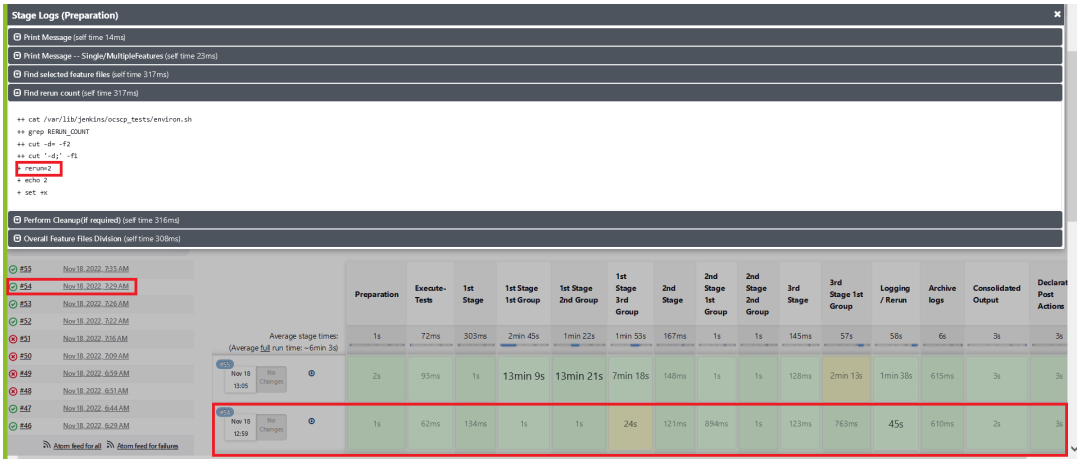
- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non-zero. If all of the test cases pass in the first run, no rerun will be initiated because the cases have already been passed. The pipeline will be green, and the status will be indicated in blue.

Figure 2-50 All of the Test cases Pass in the Initial Run



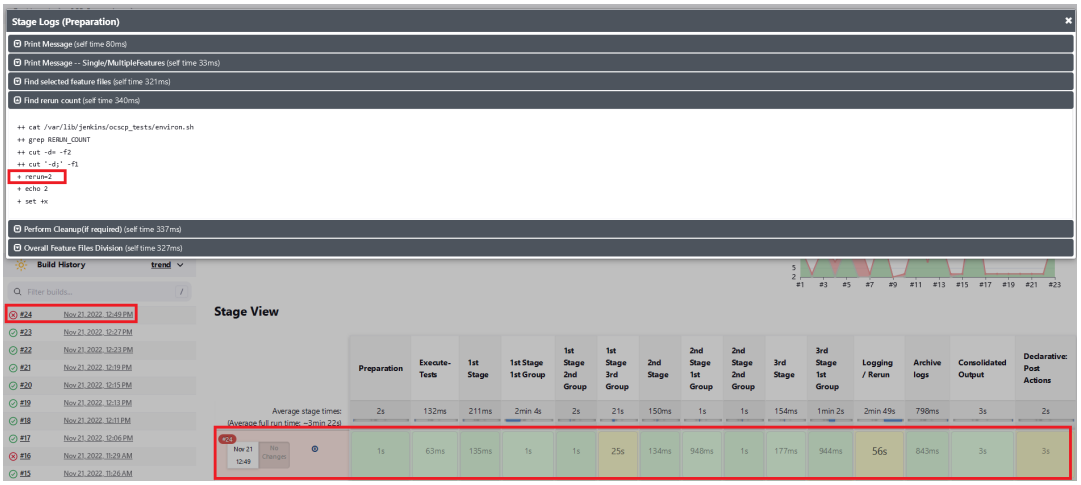
- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non-zero. If some of the test cases fail in the initial run and the remaining ones pass in one of the remaining reruns, then the initial test case execution stages will show as yellow, the remaining stages as green, and the overall pipeline status as blue.

Figure 2-51 Test Cases Fail in the Initial Run and the Remaining Ones Pass



- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non-zero. If some of the test cases fail in the initial run and the remaining ones fail in all the remaining reruns, the stages of test case execution will be shown in yellow, the remaining stages in green, and the overall pipeline status in red.

Figure 2-52 Test Cases Fail in the Initial and Remaining Reruns



- Whenever any of the multiple Behave processes that are running in the ATS are exited without completion, the stage in which the process exited and the consolidated output stage are shown as yellow, and the overall pipeline status will be yellow. Also in the consolidated output stage, near the respective stage result, the exact run in which the Behave processes exited without completion will be printed.

Figure 2-53 Stage View When Behave Process is Incomplete

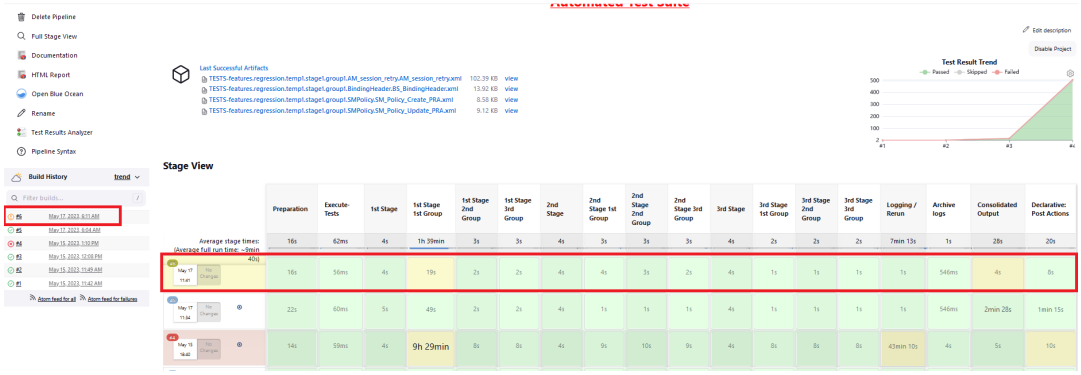
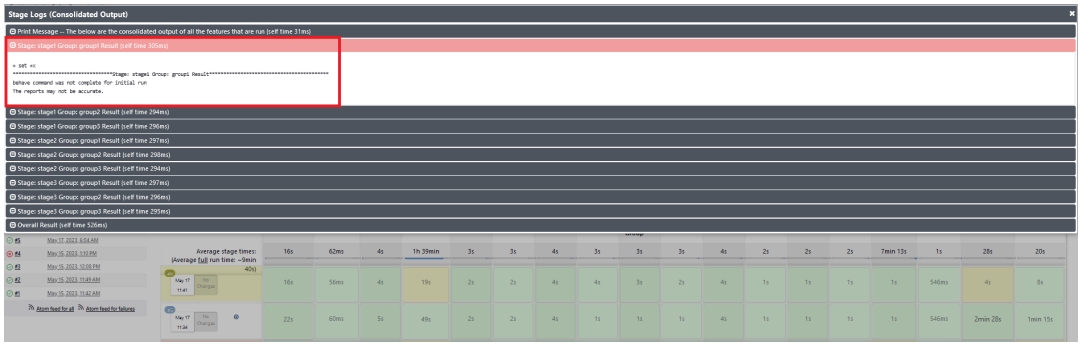


Figure 2-54 Consolidated Report for a Group When a Behave Process was Incomplete



Implementing Application Log

ATS automatically fetches the SUT Debug logs during the rerun cycle if it encounters any failures and saves them in the same location as the build console logs. The logs are fetched for the rerun time duration only using the timestamps. If, for some microservices, there are no log entries in that time duration, it does not capture them. Therefore, the logs are fetched only for the microservices that have an impact or are associated with the failed test cases.

Location of SUT Logs: `/var/lib/jenkins/.jenkins/jobs/PARTICULAR-JOB-NAME/builds/BUILD-NUMBER/date-timestamp-BUILD-N.txt`

> **Note**
>
> The file name of the SUT log is added as a suffix with the date, timestamp, and build number (for which the logs are fetched). These logs share the same retention period as build console logs, set in the ATS configuration. It is recommended to set the retention period to optimal owing to the Persistent Volume Claim (PVC) storage space availability.
>
> .

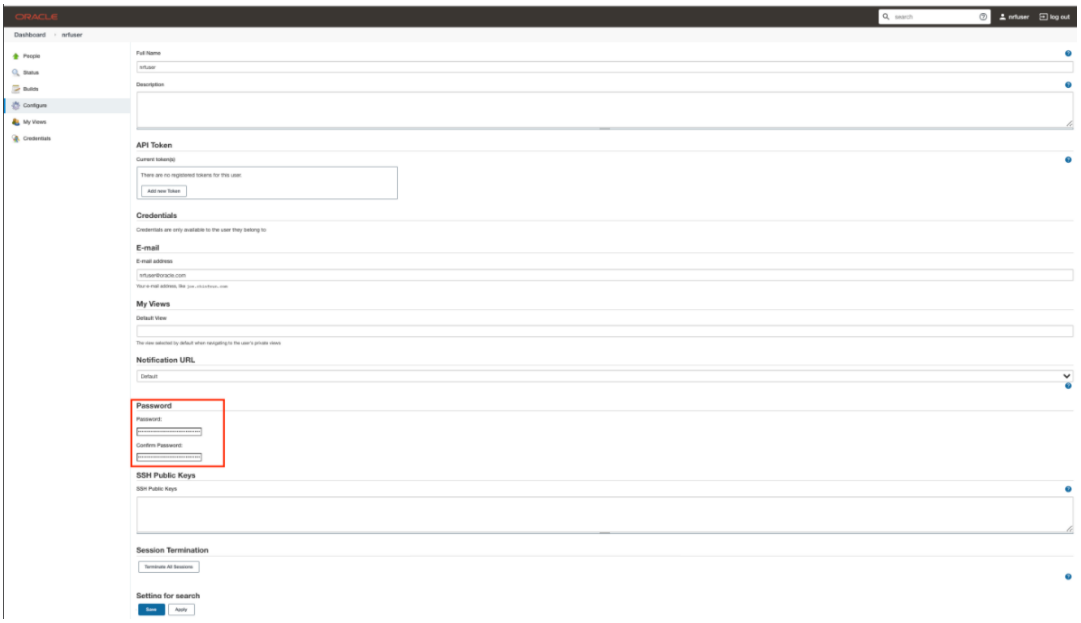
2.16 Modifying Login Password

You can log in to the ATS application using the default login credentials. The default login credentials are shared for each NF in the respective chapter of this guide.

Perform the following procedure to modify the default password:

1. Log in to the ATS application using the default login credentials. The home page of the respective NF appears.
2. Click the down arrow next to the user name.
3. Click **Configure**.
4. In the Password section, enter the new password in the **Password** and **Confirm Password** fields..

Figure 2-55 Logged-in User Details



5. Click **Save**.
   A new password is set for you.

2.17 Multiselection Capability for Features and Scenarios

ATS allows you to select and run single or multiple features and scenarios by selecting a check box for the corresponding features or scenarios.

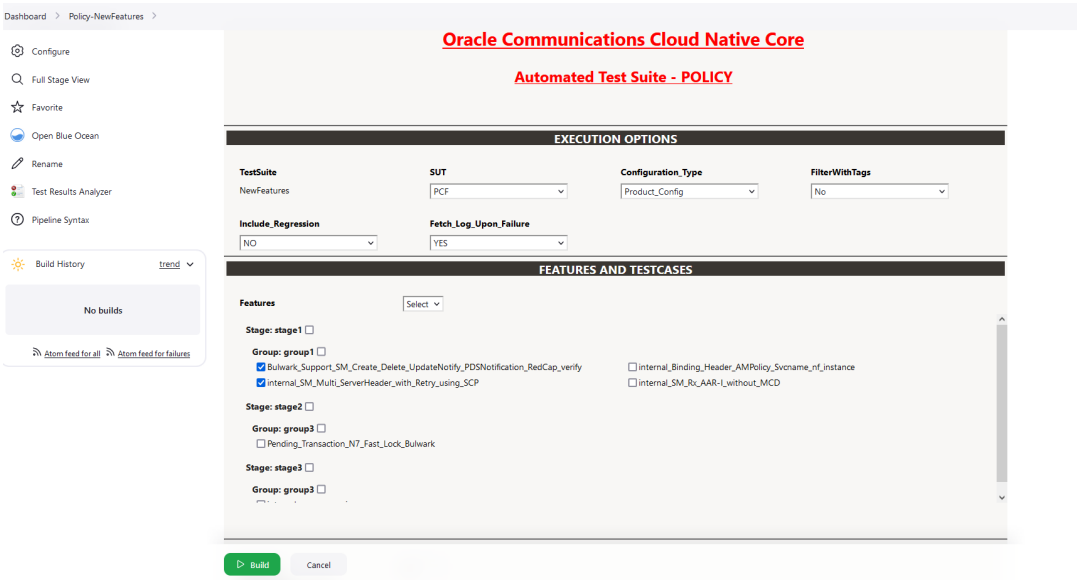
2.17.1 Feature Level Selection

Perform the following procedure to run the single or multiple features:

1. Log in to ATS using the respective <NF> login credentials.
2. On the NF home page, click any new feature or regression pipeline from where you want to run the feature.

3. In the left navigation pane, click **Build with Parameters**.
4. Scroll down to the FEATURES AND TEST CASES section.
5. Click **Select** from the **Features** drop-down.

Figure 2-56 Feature Selection



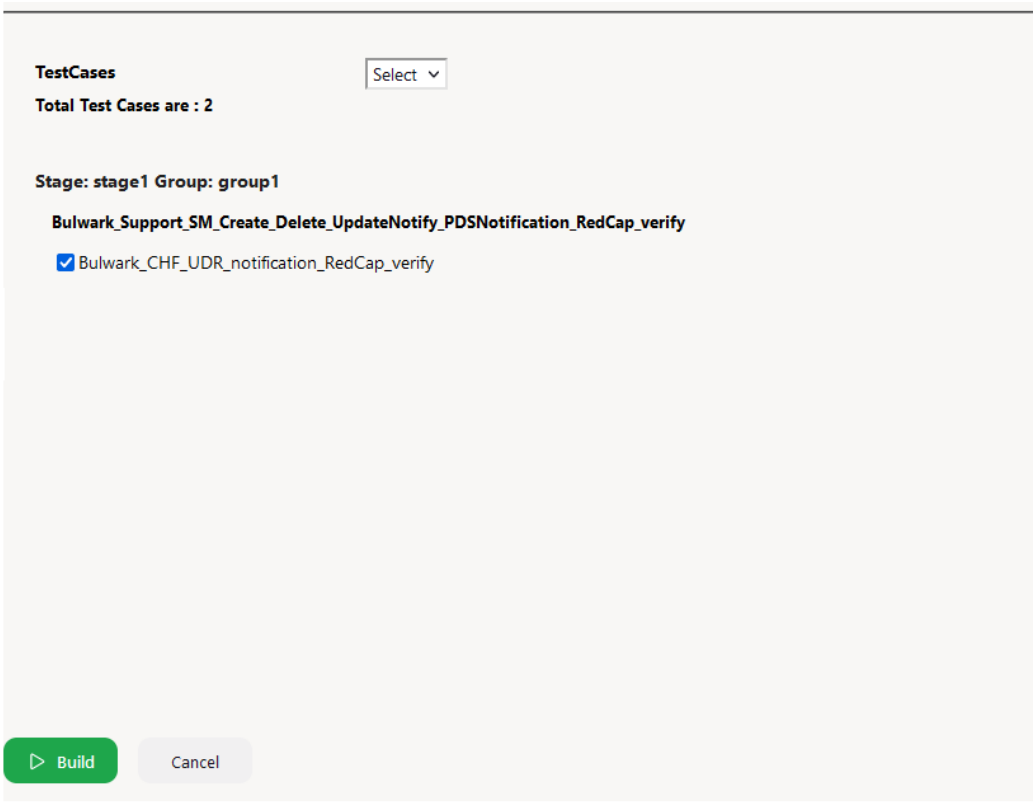
6. Select any number of features by selecting the check box for the corresponding feature you want to run from the list available for execution.
7. Click **Build**.

2.17.2 Scenario Selection

Perform the following procedure to run the single or multiple test cases or scenarios related to features:

1. Log in to ATS using the respective <NF> login credentials.
2. On the NF home page, click any new feature or regression pipeline from where you want to run the feature.
3. In the left navigation pane, click **Build with Parameters**.
4. Scroll down to the FEATURES AND TEST CASES section.
5. Click **Select** from the **Features** drop-down.
6. Select any number of features by selecting the check box for the corresponding feature to view the TestCases details mapped to each feature.
7. Click **Select** from the **TestCases** drop-down.

Figure 2-57 Scenario or Testcase Selection



8. Select the check box for the corresponding test case or scenario to run the test cases mapped to the feature.
9. Click **Build**.

2.18 Parallel Test Execution

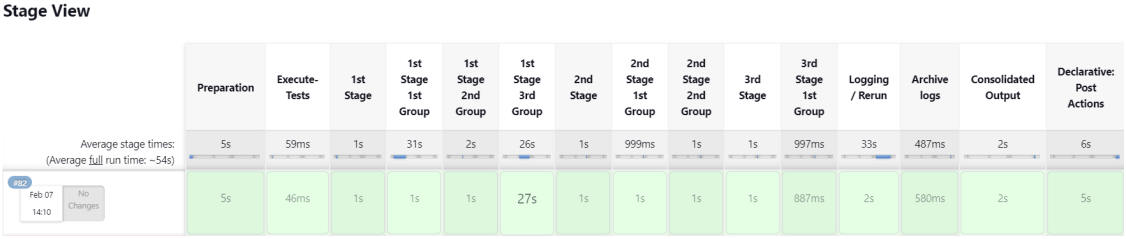
Parallel test execution allows you to perform multiple logically grouped tests simultaneously on the same System Under Test (SUT) to reduce the overall execution time of ATS.

ATS currently runs all its tests in a sequential manner, which is time-consuming. With parallel test execution, tests can be run concurrently rather than sequentially or one at a time. Test cases or feature files are now separated into different folders, such as stages and groups, for concurrent test execution. Different stages, such as stage 1, stage 2, and stage 3, run the test cases in a sequential order, and each stage has its own set of groups. Test cases or feature files available in different groups operate in parallel. When all the groups within one stage have completed their execution, only then the next stage will start the execution.

Pipeline Stage View

The pipeline stage view appears as follows:

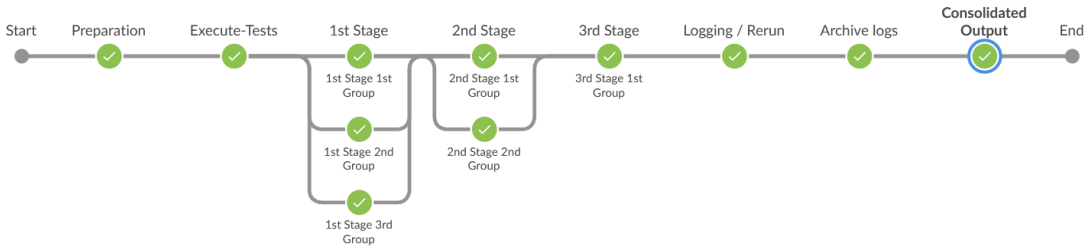
Figure 2-58 Pipeline Stage View



Pipeline Blue Ocean View

Blue Ocean is a Jenkins plugin that provides a better representation of concurrent execution with stages and groups. The pipeline blue ocean view appears as follows:

Figure 2-59 Pipeline Blue Ocean View



Merged Execution

The Merged Execution option allows you to run both new features and regression pipeline test cases together.

Perform the following procedure to run the test cases of both pipelines:

1. Log in to ATS using the respective <NF> login credentials.
   Where, <NF> indicates any 5G CNC Network Function (NF).
2. On the <NF> home page, click any new feature or regression pipeline.
3. On the displayed <NF>-NewFeature or <NF>-Regression web page, click **Build with Parameters** from the left navigation pane.
4. In the EXECUTION OPTIONS section, configure **Include_NewFeatures** and **Include_Regression** to one of the values:
   - **NO**: This is the default value. Only the test cases of the current pipeline, either Regression or NewFeatures, are run.
   - **YES**: The test case names (feature file and scenario names) of both Regression and NewFeatures are displayed on the ATS GUI, and the test cases of both pipelines are run.
5. To select and run single or multiple features of both pipelines, in the FEATURES AND TESTCASES section, select an appropriate value from the **Features** drop-down list:
   - **All**: Selects all the features of the pipeline.

- **Select**: Allows you to select the required features of the pipeline.

You can also select tags, stages, or groups as described in ATS Tagging Support and Individual Stage Group Selection.

6. Click **Build**.
Sample test case result summary report:

```
*****************************NewFeatures
Result*****************************

Final Result:-
Total-Features RUN 3, PASS 3, FAIL 0




*****************************Regression
Result*****************************
Failing scenarios:
Stage:stage1 Group:group1
  AM_Concurrency/AM_Concurrency.feature:27  AM_Concurrency_Unlocked
  AM_Concurrency_Extended_ID/AM_Concurrency_Extended_ID.feature:37
AM_Concurrency_Locked_Extended_Id
  Correlation_Header_Verification/
Correlation_Header_Verification_AM.feature:80
AM_Correlation_Generation_Verification
  Correlation_Header_Verification/
Correlation_Header_Verification_AM.feature:202
AM_Correlation_Forward_Verification
  Pending_Transaction_AM_Retry_NoRetry/
Pending_Transaction_AM_Retry_NoRetry.feature:56
Pending_Transaction_AM_Update_Notify_No_Retry
  Pending_Transaction_AM_Retry_NoRetry_Extended_ID_RedCap/
Pending_Transaction_AM_Retry_NoRetry_Extended_ID_RedCap.feature:57
Pending_Transaction_AM_Update_Notify_Retry_Extended_ID
  Revalidating_nUDR_async_nCHF_subscriptions_AM/
Revalidating_nUDR_async_nCHF_subscriptions_AM.feature:68
Revalidating_nUDR_async_nCHF_subscriptions_AM
  Subscriber_Activity_Logging_AM_ocLogId/
Subscriber_Activity_Logging_AM_ocLogId.feature:69
Subscriber_Activity_Logging_With_Subscribe_to_Notify_disabled

Initial Run:-
Features RUN 199, PASS 189, FAIL 10

1st Rerun:-
Features RUN 10, PASS 3, FAIL 7

2nd Rerun:-
Features RUN 7, PASS 0, FAIL 7

Final Result:-
Total-Features RUN 199, PASS 192, FAIL 7
```

Impact on Other Framework Features

The integration of the parallel test framework feature has an impact on the following framework features. See the following sections for more details:

- Application Log Collection
- ATS API
- Managing Final Summary Report, Build Color, and Application Log
- PCAP Log Collection

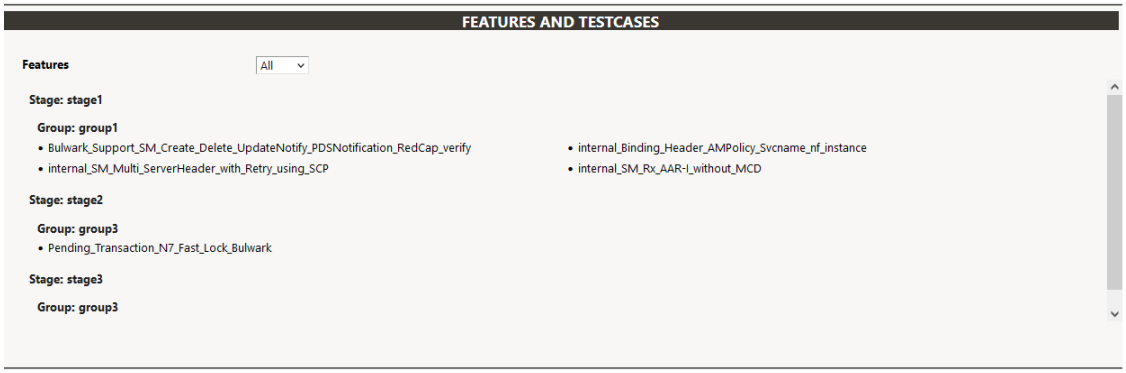
2.18.1 ATS GUI Page Changes

This section describes the changes to the ATS GUI page to trigger a build.

Changes in ATS GUI Page to Trigger a Build

The feature name, file name, and test case name are displayed under their stage and group names.

Figure 2-60 Displays Stages and Groups



2.18.2 ATS Console Log Changes

This section describes the changes to the ATS console log. The ATS console log contains logs of all the stages and groups. For more details, see Downloading or Viewing Individual Group Logs.

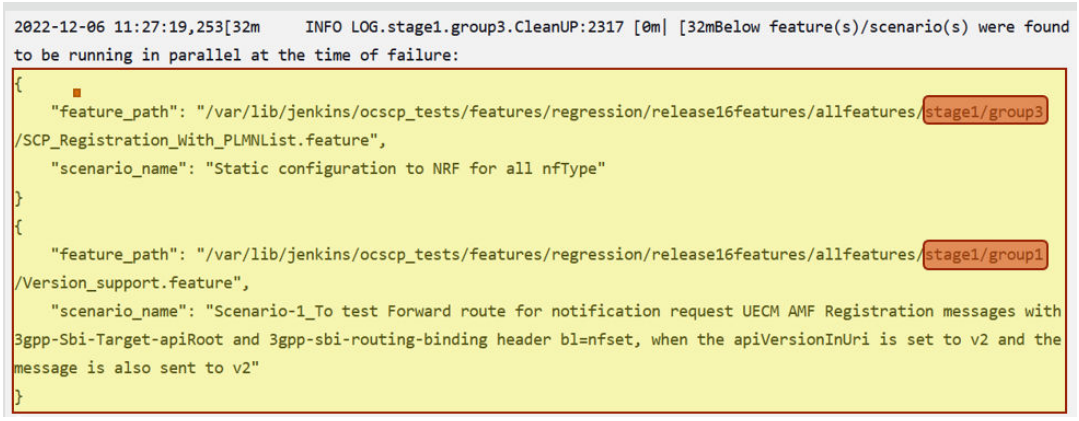
- A test case's stage and group names are listed in the logger statements for that test case.

Figure 2-61 Logger Statement

```
2022-12-06 09:34:53,762[32m     INFO LOG.stage2.group1.STEP:5453 [0m| [32m10.233.109.13[0m
2022-12-06 09:34:53,762[32m     INFO LOG.stage2.group1.STEP:5454 [0m| [32mudm1svc.scpsvc.svc.galaxy[0m
2022-12-06 09:34:53,893[32m     INFO LOG.stage2.group1.STEP:5459 [0m| [32m{"url":"/USEast/nudm-uecm/v1/imsi
```

- When a test case fails, a list of test cases running in parallel gets printed to make the debugging easier. The name of the test case and the absolute path to the feature file it belongs to are listed in this list.

Figure 2-62 Absolute Path of Feature File



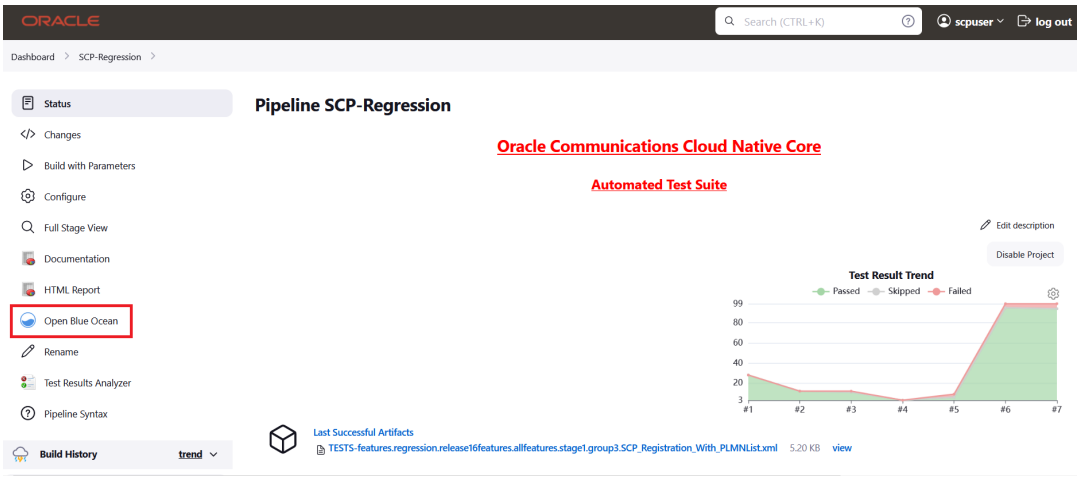
- The test result summary contains a summary for each group and an overall summary, along with the details of failing scenarios (stage-groupwise) and the total time taken by any pipeline execution. For further information, see the Managing Final Summary Report, Build Color, and Application Log.

2.18.3 Downloading or Viewing Individual Group Logs

To download individual group logs:

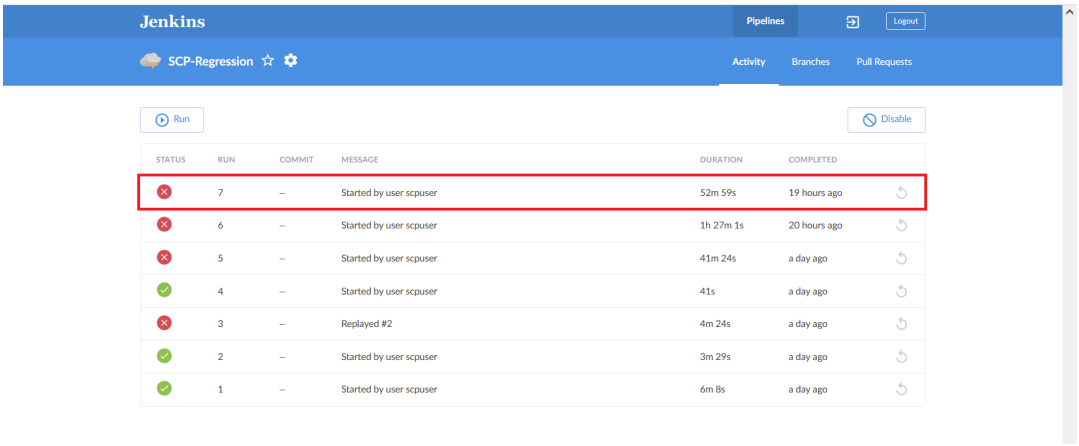
1. On the Jenkins pipeline page, click **Open Blue Ocean** in the left navigation pane.

Figure 2-63 Jenkins Pipeline Page



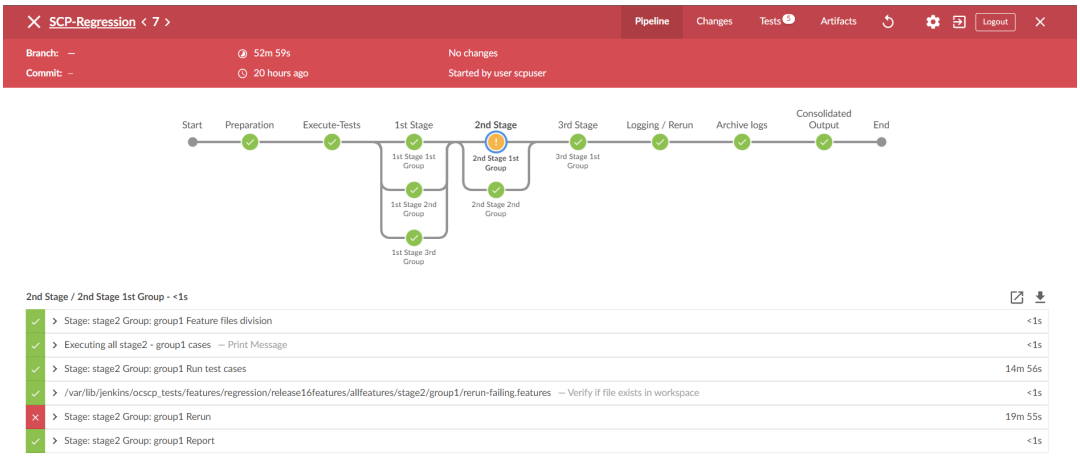
2. Click the desired build row on the Blue Ocean page.

Figure 2-64 Run the Build



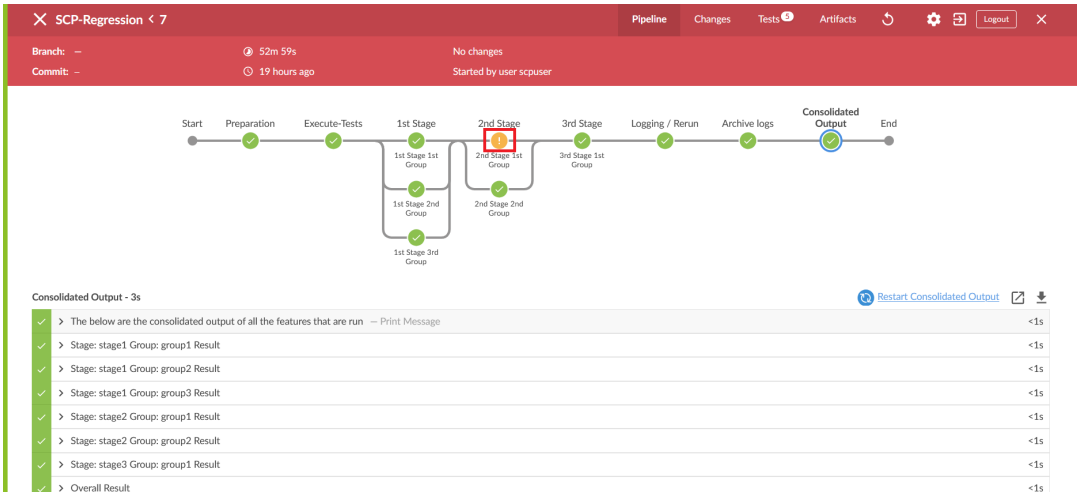
3. The selected build appears. The diagram displays the order in which the different stages, or groups, are executed.

Figure 2-65 Stage Execution



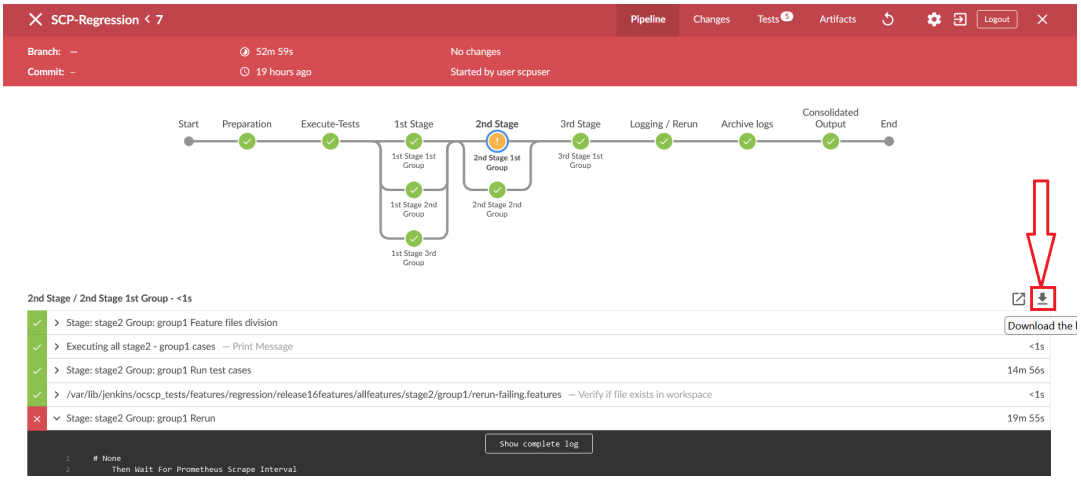
4. Click the desired group to download the logs.

Figure 2-66 Executed Groups



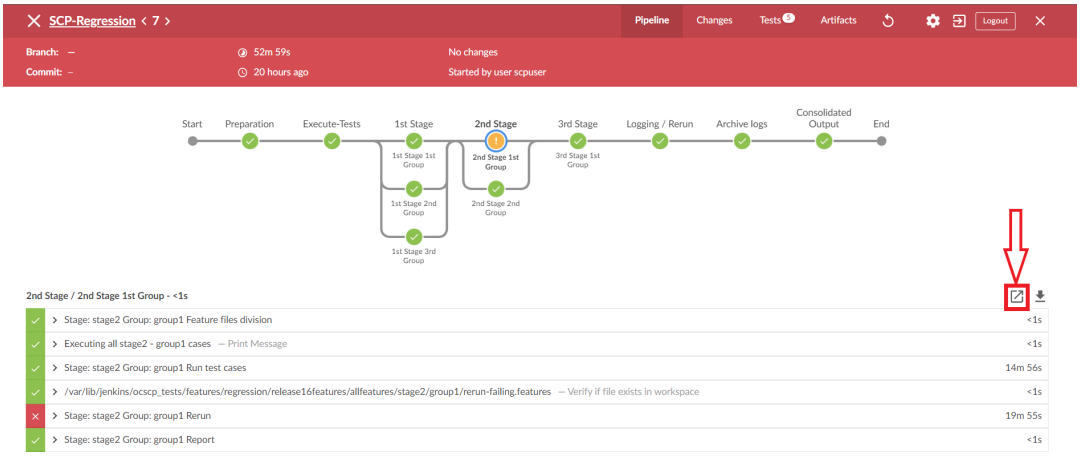
5. Click the **Download** icon on the bottom right of the pipeline. The log for the selected group is downloaded to the local system.

Figure 2-67 Download Logs



6. To view the log, click the **Display Log** icon. The logs are displayed in a new window.

Figure 2-68 Display Logs



Viewing Individual Group Logs without using Blue Ocean

There are two alternate ways to view individual group logs:

- Using Stage View

  1. On the Jenkins pipeline page, hover the cursor over the group in stage view to view the logs.
  2. Click the label **Logs**" which appears in a pop-up.
     There will be a new pop-up window. It contains many rows, where each row corresponds to the execution of one Jenkins step.
  3. Click the row labeled **Stage: stage_name>."Group: <group_name>** Run test cases to view the log for this group's execution.

4. Click the row labeled **Stage: stage_name>." "group_name> Rerun** to display the rerun logs.

- Using Pipeline Steps Page

1. On the Jenkins pipeline page, click the desired build number under the **Build History** drop down.
2. Click the **Pipeline Steps** button on the left pane.
   A table with columns for step, arguments, and status appears.
3. Under the Arguments column, find the label for the desired stage and group.
4. Click the step with the label **Stage: <stage_name>Group: <group_name>**.
5. Run test cases under it, or click the **Console output** icon near the status to view the log for this group execution.
6. To see rerun logs, find the step with the label **Stage: <stage_name> Group: <group_name>Rerun** under it.

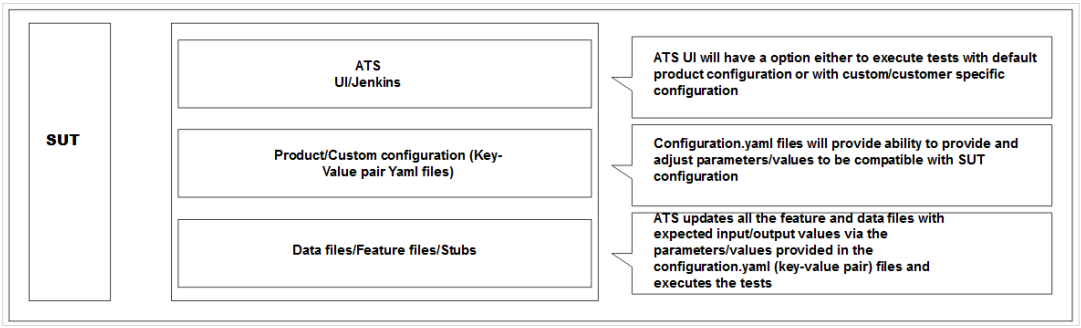
2.19 Parameterization

This feature allows you to provide or adjust values for the input and output parameters needed for the test cases to be compatible with the SUT configuration. You can update or adjust the key-value pair values in the `global.yaml` and `feature.yaml` files for each of the feature files so that they are compatible with SUT configuration. In addition to the existing custom test case folders (Cust New Features, Cust Regression, and Cust Performance), this feature enables folders to accommodate custom data, default product configuration, and custom configuration. You can maintain multiple versions or copies of the custom data folder to suit varied or custom SUT configurations. With this feature, the ATS GUI has the option to either execute test cases with the default product configuration or with a custom configuration.

This feature enables you to perform the following tasks:

- Define parameters and assign or adjust values to make them compatible with SUT configuration.
- Execute test cases either with default product configurations or custom configurations and multiple custom configurations to match varied SUT configurations.
- Assign or adjust values for input or output parameters through custom or default configuration yaml files (key-value pair files).
- Define or adjust the input or output parameters for each feature file with its corresponding configuration.
- Create and maintain multiple configuration files to match multiple SUT configurations.

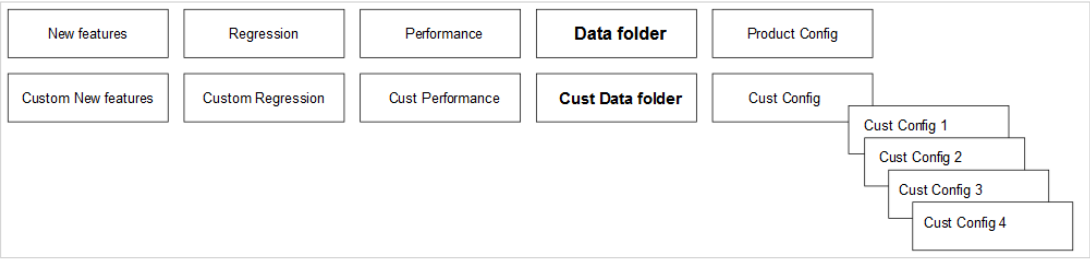
Figure 2-69 SUT Design Summary



In the folder structure:

- The Product Config folder contains default product configuration files (feature-wise yaml per key-value pair), which are compatible with default product configuration.
- New features, Regression and Performance, Data folder, and Product Config folders are replicated or copied into custom folders and delivered as part of the ATS package in every release.
- You can customize custom folders by:
  - Removing test cases not needed or as appropriate for your use.
  - Adding new test cases as needed or as appropriate for your use.
  - Removing or adding data files in the cust_data folder or as appropriate for your use.
  - Adjusting the parameters or values in the key-value pair per yaml file in the custom config folder for test cases to run or pass with a custom configured SUT.
- The product folders are always intact (unchanged) and you can update the Custom folders
- You can maintain multiple copies of Custom Configurations and bring them to use as needed or as appropriate for the SUT configuration.

Figure 2-70 Folder Structure



2.19.1 Running Test Cases

Enable

Rename or copy the Cust Config [1/2/3/N] folder to the Cust Config folder in order for ATS to run the test cases with a specific custom configuration. When the option to run test cases with custom configuration is chosen, it always points to the Cust Config folder.

To Run ATS Test Cases

ATS has the option to run test cases with default or custom configuration.

- If custom configuration is selected, then test cases from custom folders are populated on the ATS UI, and custom configuration is applied to them through the key-value pair per yaml file defined or present in the "Cust Config" folder.
- If product configuration is selected, then the test cases from product folders are populated on the ATS UI, and product configuration is applied to them through key-value pairs per yaml file defined or present in the Product Config folder.

Figure 2-71 ATS Execution Flow

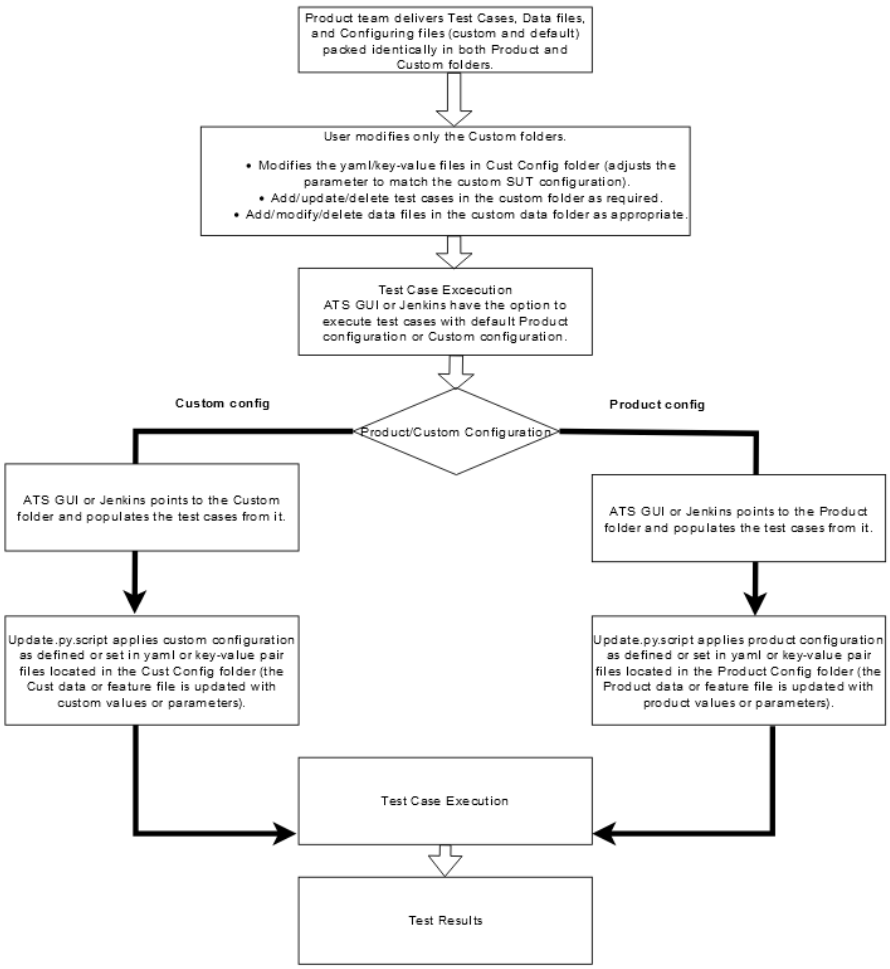
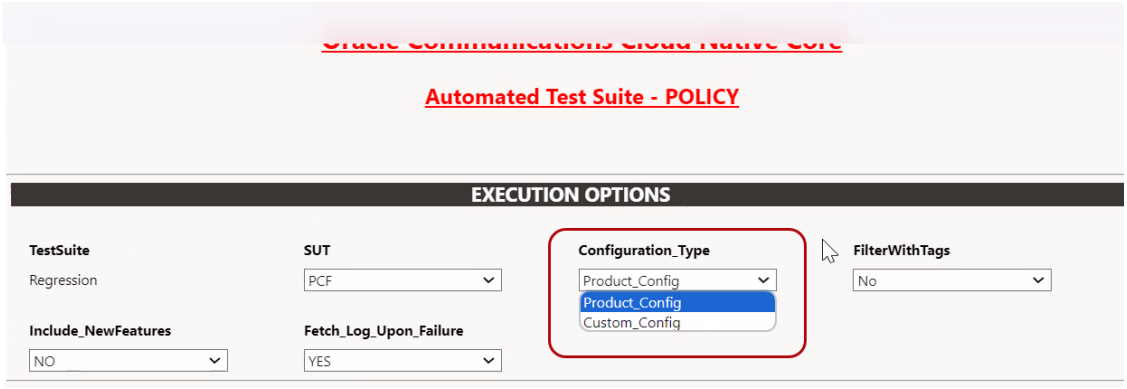


Figure 2-72 Sample: Configuration_Type



2.20 PCAP Log Collection

PCAP Log Collection allows collecting the NF, SUT, or PCAP logs from the debug tool sidecar container. This feature can be integrated and delivered as a standalone or along with the Application Log Collection feature. For information, see Application Log Collection.

PCAP Log Integration

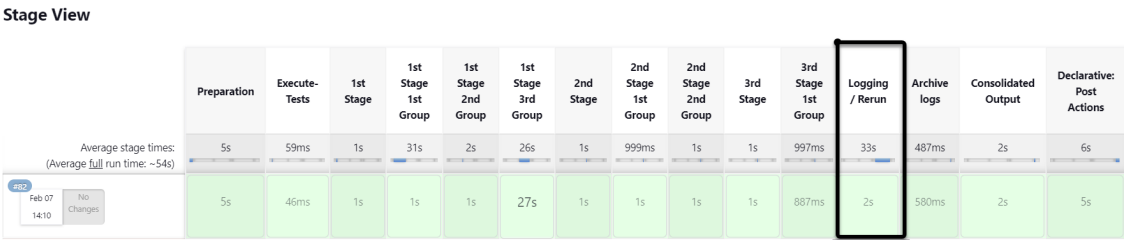
1. The Debug tool should be enabled on SUT Pods while deploying the NF. The name of the Debug container must be "tools".
   For example, in SCP, the debug tool should be enabled for all the SCP microservice pods.
2. Update the following parameters in the values.yaml file, under the resource section, with ATS minimum resource requirements:
   a. CPU: 3
   b. memory: 3Gi
3. On the home page, click any new feature or regression pipeline.
4. In the left navigation pane, click **Build with Parameters**.
5. Select **YES** from the **Fetch_Log_Upon_Failure** drop-down menu.
6. If option **Log_Type** is available, select value **PcapLog [Debug Container Should be Running]** for it.
7. Select **PcapLog [Debug Container Should be Running]** to activate PCAP Log Collection in ATS-NF.
8. After the build execution is complete, go into the ATS pod, then navigate to below path to find the pcaplogs: `.jenkins/jobs/<Pipeline Name>/builds/<build number>/`
   For example, `.jenkins/jobs/SCP-Regression/builds/5/`

   Pcaplogs is present in zip form. Unzip it to get the log files.

2.20.1 Application Log Collection and Parallel Test Execution Integration

A new stage,"Logging/Rerun", has been added at the end of the Execute-Tests stage to collect rerun logs, such as applog and PCAP logs, by running the failed test cases in a sequence.

Figure 2-73 Logging/Rerun new stage



If the `Fetch_Log_Upon_Failure` parameter is set to YES and if any test case fails in the initial run, then:

- The failed test case reruns and log collection start in the Logging/Rerun stage after the initial run is completed for all the test cases.
- The logs from the initial execution are collected, but they might be incorrect.
- Even if the `rerun` parameter is set to 0, the failed test case reruns in the Logging/Rerun stage and the log is collected.

> **Note**
>
> Not applicable for all the NFs.

- If the `Fetch_Log_Upon_Failure` parameter is set to NO and if any test case fails in the initial run, then the failed test case rerun starts in the same stage after the initial execution is over for all the test cases in its group.

2.21 Persistent Volume for 5G ATS

The Persistent Volume (PV) feature allows ATS to retain historical build execution data, test cases, and ATS environment configurations.

ATS Packaging When Using Persistent Volume

- **Without the Persistent Volume option:** ATS package includes an ATS image with test cases.
- **With Persistent Volume option:** ATS package includes the ATS image and test cases separately. The new test cases are provided between the releases.
  To support both with and without Persistent Volume options, test cases and execution job data are packaged in the ATS image as well as a tar file.

2.21.1 Processing Flow

First Time Deployment

Initially, when you deploy ATS, for example, the PI-A ATS pod, you use PVC-A, which is provisioned and mounted to the PI-A ATS pod. By default, PVC-A is empty. So, you have to copy the data (ocslf_tests and jobs folders) from the PI-A tar file to the pod after the pod is up and running. Then restart the PI-A pod. At this point, you can change the number of build logs to maintain in the ATS GUI.

Subsequent Deployments

When you deploy ATS for the subsequent time, for example, in a PI-B ATS pod, you use PVC-B, which is provisioned and mounted to the PI-B ATS pod. By default, the PVC-B is empty, and you have to copy the data (ocslf_tests and jobs folders) from the PI-B tar file to the pod after the pod is up and running. At this point, copy all the necessary changes to the PI-B pod from the PI-A pod and restart the PI-B pod. You can change the number of build logs to maintain in the ATS GUI. After updating the number of builds, you can delete the PI-A pod and continue to retain the PVC-A. If you do not want backward porting, you can delete PVC-A.

2.21.2 Deploying Persistent Volume

Preinstallation Steps for Non-OCI setup

1. Before deploying Persistent Volume, create a PVC in the same namespace where you have deployed ATS. You have to provide values for the following parameters to create a PVC:
   - PVC Name
   - Namespace
   - Storage Class Name
   - Size of the PV
2. Run the following command to create a PVC:

```
kubectl apply -f - <<EOF

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <Please Provide the PVC Name>
  namespace: <Please Provide the namespace>
  annotations:
spec:
  storageClassName: <Please Provide the Storage Class Name>
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <Please Provide the size of the PV>
EOF
```

> **Note**
>
> It is recommended to suffix the PVC name with the release version to avoid confusion during subsequent releases. For example: ocats-slf-pvc-1.9.0

The output of the above command with parameters is as follows:

```
[cloud-user@atscne-bastion-1 templates]$ kubectl apply -f - <<EOF
>
> apiVersion: v1
> kind: PersistentVolumeClaim
> metadata:
```

```
>     name: ocats-slf-1.9.0-pvc
>     namespace: ocslf
>     annotations:
> spec:
>     storageClassName: standard
>     accessModes:
>         - ReadWriteOnce
>     resources:
>         requests:
>             storage: 1Gi
> EOF
```

The persistentvolumeclaim/ocats-slf-1.9.0-pvc is created.

3. To verify whether PVC is bound to PV and is available for use, run the following command:

```
kubectl get pvc -n <namespace used for pvc creation>
```

The output of the above command is as follows:

Figure 2-74 Verifying PVC

```
[cloud-user@atscne-bastion-1 templates]$ kubectl get pvc -n ocslf
NAME                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES STORAGECLASS   AGE
ocats-slf-1.9.0-pvc   Bound    pvc-1ee50daf-2eb7-4289-b541-51c111270513   1Gi        RWO          standard       8s
policy-pvc-test       Bound    pvc-d346d5c4-129e-4f13-8bd0-a7b7555e0e2f   1Gi        RWO          standard       31d
slf-pvc               Bound    pvc-0b0bce8c-b856-4e0e-b4d8-0f981250adbd   1Gi        RWO          standard       12d
```

Check that the STATUS is **Bound** and that the rest of the parameters, such as NAME, CAPACITY, ACCESS MODES, STORAGECLASS, and so on, are the same as specified in the PVC creation command.

> **Note**
>
> Do not proceed further if there is any issue with PVC creation. Contact your administrator to create a PV.

4. After creating persistent volume, change the following parameters in the values.yaml file to deploy persistent volume.
   - Set the **PVEnabled** parameter to "true".
   - Provide the value for the **PVClaimName** parameter. The PVClaimName value should be the same as the value used to create a PVC.

Preinstallation Steps for OCI setup

1. Before deploying persistent volumes, create a PVC in the same namespace where you have deployed ATS. You have to provide values for the following parameters to create a PVC:
   - PVC Name
   - Namespace
   - Storage Class Name
   - Size of the PV

2. Run the following command to create a PVC:

```
kubectl apply -f - <<EOF

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <Please Provide the PVC Name>
  namespace: <Please Provide the namespace>
  annotations:
spec:
  storageClassName: <Please Provide the Storage Class Name>
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <Please Provide the size of the PV>
EOF
```

3. When creating Persistent Volume Claim in OCI, add the storageClassName "oci-bv" and maintain the 50 gigabyte storage value.

> **Note**
>
> The minimum allowed value for parameter "resources.requests.storage" is 50Gi. If the storage is set to any value less than 50Gi, it will be ignored.

For example, the output of the above command with parameters is as follows:

```
[cloud-user@atscne-bastion-1 templates]$ kubectl apply -f - <<EOF
>
> apiVersion: v1
> kind: PersistentVolumeClaim
> metadata:
>     name: ocats-slf-1.9.0-pvc
>     namespace: ocslf
>     annotations:
> spec:
>     storageClassName: oci-bv
>     accessModes:
>         - ReadWriteOnce
>     resources:
>         requests:
>             storage: 50Gi
> EOF
```

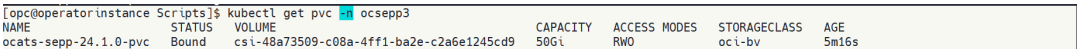
The persistentvolumeclaim/ocats-slf-1.9.0-pvc is created.

4. To verify whether PVC is bound to PV and is available for use, run the following command:

```
kubectl get pvc -n <namespace used for pvc creation>
```

The output of the above command is as follows:

Figure 2-75 Verifying PVC



5. After creating persistent volume, the STATUS will be Pending and when the ATS pod is deployed with that PVC, its status will become "BOUND".
6. Change the following parameters in the values.yaml file to deploy persistent volume.
   - Set the **PVEnabled** parameter to "true".
   - Provide the value for the **PVClaimName** parameter. The PVClaimName value should be the same as the value used to create a PVC.

Postinstallation Steps

1. After deploying ATS, copy the <nf_main_folder> and <jenkins jobs> folders from the tar file to their ATS pod, and then restart the pod as a one-time activity.
   a. Run the following command to extract the tar file:

      ```
      ocats-<nf_name>-data-<release-number>.tgz
      ```

      > **Note**
      >
      > The ats_data.tar file is the name of the tar file containing <nf_main_folder> and jobs folders. It can be different for different NFs.

   b. Run the following set of commands to copy the required folders:

      ```
      kubectl cp ats_data/jobs <namespace>/<pod-name>:/var/lib/
      jenkins/.jenkins/
      ```

      ```
      kubectl cp ats_data/<nf_main_folder> <namespace>/<pod-name>:/var/lib/
      jenkins/
      ```

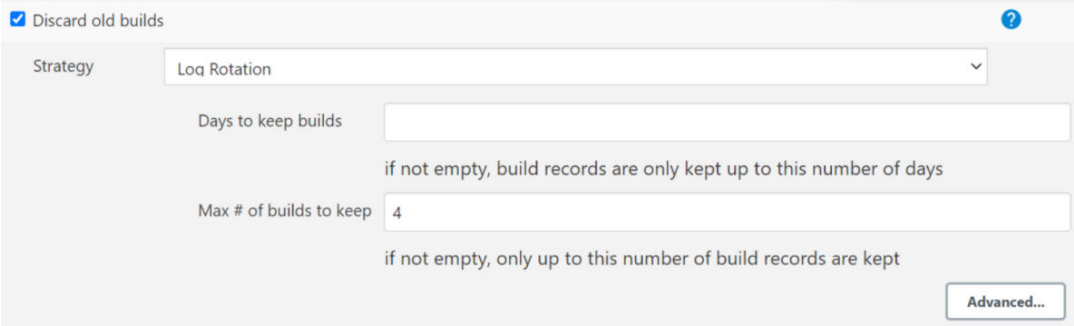
   c. Run the following command to restart the pods as one-time activity:

      ```
      kubectl delete po <pod-name> -n <namespace>
      ```

      > **Note**
      >
      > Before running the following command, copy the changes done on the new release pod from the old release pod using the `kubectl cp` command. **[Applicable in the case of subsequent deployment only]**

2. When the pod is up and running, log in to the ATS GUI and go to your NF specific pipeline. Click **Configure** in the left navigation pane. The **General** tab appears. Configure the **Discard old Builds** option. This option allows you to configure the number of builds you want to retain in the persistent volume.

Figure 2-76 Discard Old Builds



> **Note**
>
> It is recommended to configure this option. If you do not input a value for this option, the application will take into account all builds, which could be a large number, and will completely consume the Persistent Volume.

2.21.3 Backward Porting

The following deployment steps apply to the old release of PVC-supported ATS Pod.

Prerequisite: You should have the old PVC that contains the old release of POD data.

> **Note**
>
> This procedure is for backward porting purposes only and should not be considered a subsequent release of the POD deployment procedure.

The deployment procedure for the old release PVC-supported ATS pod is the same; however, while deploying the ATS pod, you have to update the values.yaml file with the following:

- Change the **PVEnabled** parameter to "true".
- Provide the name of the old PVC as the value for parameter **PVClaimName**.

2.22 Single Click Job Creation

With the help of Single Click Job Creation feature, ATS users can easily create a job to run TestSuite with a single click.

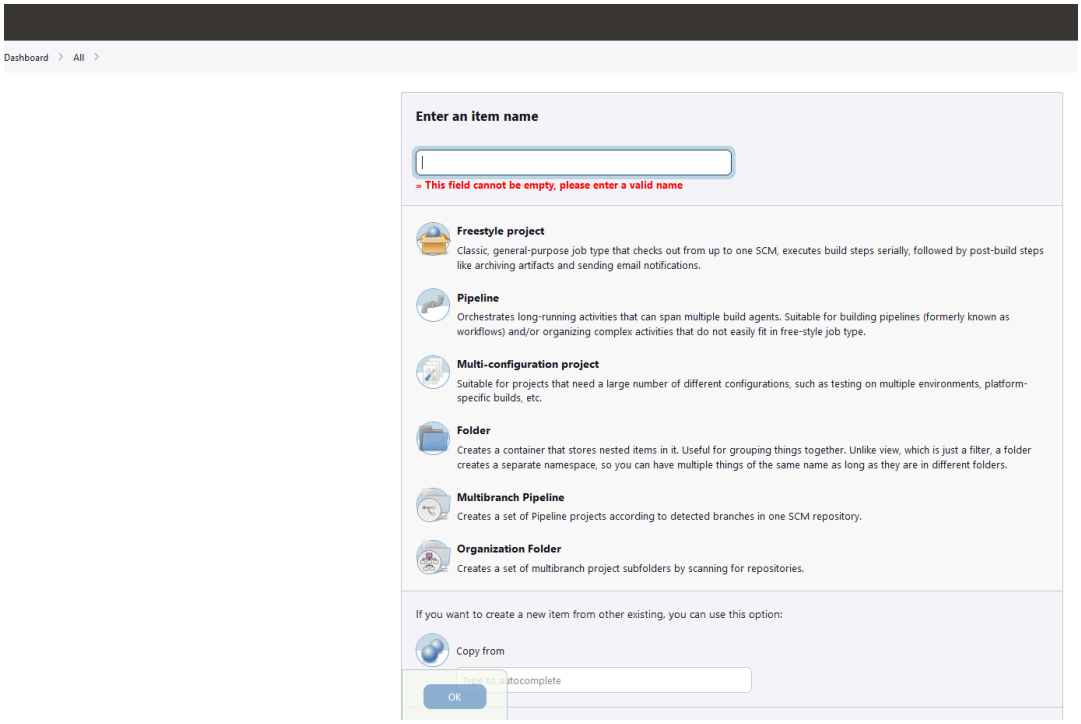
2.22.1 Configuring Single Click Job

Prerequisite: The network function specific user should have 'Create Job' access.

Perform the following procedure to configure the single-click feature:

1. Log in to ATS using network function specific log-in credentials..
2. Click **New Item** in the left navigation pane of the ATS application. The following page appears:

Figure 2-77 New Item Window



3. In the **Enter an item name** text box, enter the job name. Example: <NF-Specific-name>-NewFeatures.
4. In the **Copy from** text box, enter the actual job name for which you need single-click execution functionality. Example: <NF-Specific-name>-NewFeatures.
5. Click **OK**. You are automatically redirected to edit the newly created job's configuration.
6. Under the **General** group, deselect the **This Project is Parameterised** option.
7. Under the **Pipeline** group, make the corresponding changes to remove the 'Active Choice Parameters' dependency.
8. Provide the default values for the **TestSuite**, **SUT**, **Select_Option**, **Configuration_Type**, and other parameters, as required, on the **BuildWithParameters** page.
   Example: Pipeline without Active Choice Parameter Dependency

```
node ('built-in'){
    //a = SELECTED_NF    b = PCF_NAMESPACE        c = PROMSVC_NAME       d
= GOSTUB_NAMESPACE
    //e = SECURITY       f = PCF_NFINSTANCE_ID   g = POD_RESTART_TIME   h
= POLICY_TIME
    //i = NF_NOTIF_TIME  j = RERUN_COUNT         k =INITIALIZE_TEST_SUITE
l = STUB_RESPONSE_TO_BE_SET
    //m = POLICY_CONFIGURATION_ADDITION          n = POLICY_ADDITION
o = NEXT_MESSAGE
    //p = PROMSVCIP     q = PROMSVCPORT         r = TIME_INT_POD_DOWN    s
= POD_DOWN_RETRIES
    //t = TIME_INT_POD_UP   u = POD_UP_RETRIES  v = ELK_WAIT_TIME   w =
ELK_HOST
    //x = ELK_PORT  y = STUB_LOG_COLLECTION z = LOG_METHOD A =
enable_snapshot B = svc_cfg_to_be_read C = PCF_API_ROOT
```

```
    //Description of Variables:

    //SELECTED_NF : PCF
    //PCF_NAMESPACE : PCF Namespace
    //PROMSVC_NAME : Prometheus Server Service name
    //GOSTUB_NAMESPACE : Gostub namespace
    //SECURITY : secure or unsecure
    //PCF_NFINSTANCE_ID : nfInstanceId in PCF application-config config map
    //POD_RESTART_TIME : Greater or equal to 60
    //POLICY_TIME : Greater or equal to 120
    //NF_NOTIF_TIME : Greater or equal to 140
    //RERUN_COUNT : Rerun failing scenario count
    //TIME_INT_POD_DOWN : The interval after which we check the POD status
if its down
    //TIME_INT_POD_UP : The interval after which we check the POD status
if its UP
    //POD_DOWN_RETRIES : Number of retry attempt in which will check the
pod down status
    //POD_UP_RETRIES : Number of retry attempt in  which will check the
pod up status
    //ELK_WAIT_TIME : Wait time to connect to Elastic Search
    //ELK_HOST : Elastic Search HostName
    //ELK_PORT : Elastic Search Port
    //STUB_LOG_COLLECTION : To Enable/Disable Stub logs collection
    //LOG_METHOD : To select Log collection method either elasticsearch or
kubernetes
    //enable_snapshot: Enable or disable snapshots that are created at the
start and restored at the end of each test run
    //svc_cfg_to_be_read: Timer to wait for importing service
configurations
    //PCF_API_ROOT: PCF_API_ROOT information to set Ingress gateway
service name and port
    withEnv([
    'TestSuite=NewFeatures',
    'SUT=PCF',
    'Select_Option=All',
    'Configuration_Type=Custom_Config'
    ]){
    sh '''
        sh /var/lib/jenkins/ocpcf_tests/preTestConfig-NewFeatures-PCF.sh \
        -a PCF \
        -b ocpcf \
        -c occne-prometheus-server \
        -d ocpcf \
        -e unsecure \
        -f fe7d992b-0541-4c7d-ab84-c6d70b1b0123 \
        -g 60 \
        -h 120 \
        -i 140 \
        -j 2 \
                -k 0 \
                -l 1 \
                -m 1 \
                -n 15 \
                -o 1 \
                -p occne-prometheus-server.occne-infra\
```

```
                    -q 80\
                    -r 30\
                    -s 5\
                    -t 30\
                    -u 5\
                    -v 0\
                    -w occne-elastic-elasticsearch-master.occne-infra\
                    -x 9200\
                    -y yes\
                    -z kubernetes\
                    -A no\
                    -B 15\
                    -C ocpcf-occnp-ingress-gateway:80\

    '''
    load "/var/lib/jenkins/ocpcf_tests/jenkinsData/Jenkinsfile-Policy-
NewFeatures"
   }
}
```

9. Click **Save**. The ATS application is ready to run TestSuite with 'SingleClick' using the newly created job.

2.23 Support for ATS Deployment in OCI

Oracle Cloud Infrastructure (OCI) is a set of complementary cloud services that enable you to build and run a range of applications and services in a highly-available, consistently high-performance environment. OCI offers powerful computing capabilities and storage capacity in a flexible virtual network that can be accessed from your on-premises network.

OCI infrastructure consists of Compartments, Network Load balancer, Bastion Host, Dynamic Routing Gateway (DRG), Remote Peering Connection (RPC), Service Gateway, Internet Gateway, and Oracle Kubernetes Engine (OKE) cluster, all in an OKE VCN.

ATS deployment is also supported in OCI.

2.23.1 Accessing ATS GUI in OCI

The following two ways are supported to access the ATS GUI in OCI:

Using Loadbalancer IP

1. Add proper Ingress or Egress security rules for ports (8080/8443) and ATS service Node port in loadbalancer (nf_lb_subnet) and Node subnet (nf_node_subnet). To add ingress and egress security rules, see the Adding Ingress and Egress Rules to Access the OCI Console.
2. Insert the following annotations under the **Metadata** section in ATS service object to assign an external IP (Loadbalancer IP):

```
oci-network-load-balancer.oraclecloud.com/security-list-management-mode:
None
oci.oraclecloud.com/load-balancer-type: nlb
```

> **Note**
>
> This can be done by editing and saving ATS service object after ATS deployment: For example,
>
> ```
> kubectl edit svc ats-service-name -n ats-namespace
> ```

3. Access the GUI using URL: `<http/https>://<Loadbalancer IP>:<8080/8443>`

> **Note**
>
> - To open ATS GUI with https (TLS enabled), Load balancer IP must be known before deployment and listed down in the alt_names section of the ssl.conf file while generating the application or client certificate. Otherwise, the ATS GUI will open, but it will show "Not Secure" in the browser address bar. For more information, see the Support for Transport Layer Security section.
> - The assignment of Loadbalancer IP to the ATS service is subject to availability. If the Loadbalancer IP is not assigned to the ATS service even after applying the required annotations, try to debug on the OCI side.

Using Tunneling

1. Add an ingress security rule for the node subnet (nf_node_subnet) to allow TCP traffic on all ports from the operator subnet. To add ingress and egress security rules, see the Adding Ingress and Egress Rules to Access the OCI Console section.
2. Run the following command from a bash terminal on your local PC:

   ```
   ssh -f -N -i <operator instance private key> -o StrictHostKeyChecking=no -
   o ProxyCommand="ssh -i <bastion private key> -o StrictHostKeyChecking=no -
   W %h:%p <bastion username>@<bastion IP>" <operator instance
   username>@<operator instance IP> -L <local PC port>:<Worker Node IP>:<ATS
   NodePort> -o ServerAliveInterval=60 -o ServerAliveCountMax=300
   ```

3. Access the GUI using URL: `<http/https>://<localhost/127.0.0.1>:<local PC port>` For example,`ssh -f -N -i id_rsa -o StrictHostKeyChecking=no -o ProxyCommand="ssh -i id_rsa -o StrictHostKeyChecking=no -W %h:%p opc@129.287.66.123" opc@10.1.76.7 -L 5009:10.9.60.118:32071 -o ServerAliveInterval=60 -o ServerAliveCountMax=300`

   Here, ATS GUI URLis http://localhost:5009

2.23.2 Adding Ingress and Egress Rules to Access the OCI Console

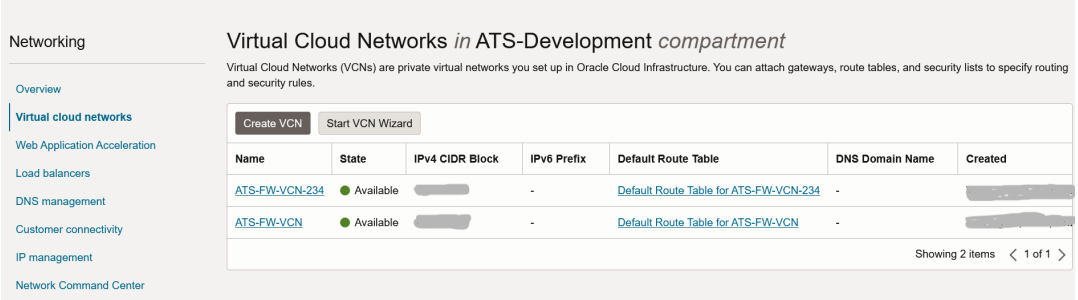
Perform the following procedure to add ingress and egress rules:

1. Log in to the OCI Console using your login credentials.
2. On the OCI Console Dashboard, click **Navigation menu**.

Figure 2-78 Navigation menu

3. Click **Networking**, and then click **Virtual cloud networks** on the preview pane.

Figure 2-79 Virtual cloud networks

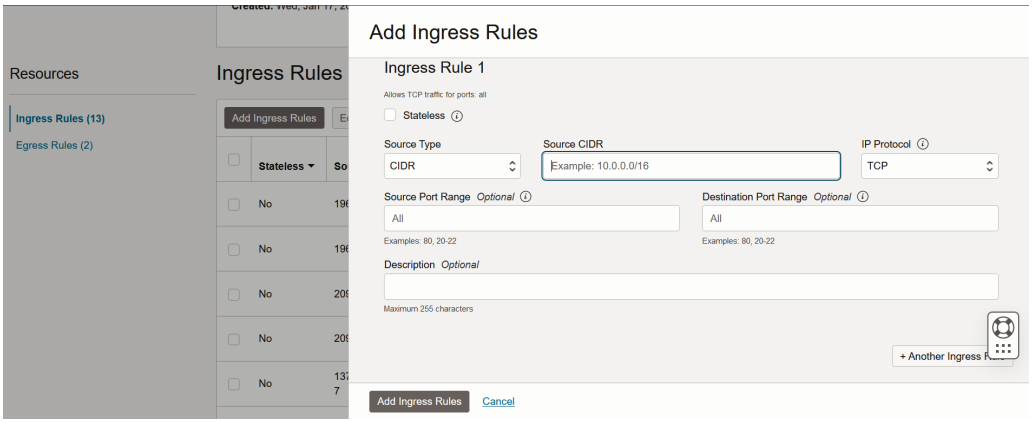


4. Click the required OKE cluster VCN.
   A list of all subnets in that VCN is displayed.
5. Save the value of the "IPv4 CIDR Block" field for **nf_lb_subnet**.
   This information can be used in the subsequent steps as value of `<CIDR for Load balancer subnet>`.
6. Save the value of the "IPv4 CIDR Block" field for **nf_node_subnet**.
   This information can be used in the subsequent steps as value of `<CIDR for Node subnet>`.

Updating Security List for Node Subnet

1. To update security list for Node subnet, do the following:
   a. Navigate to **nf_node_subnet**, and then to **nf_node_security_list**.
   b. Click **Add Ingress Rules**.

Figure 2-80 Add Ingress Rules



   c. Add ingress rule if it is unavailable to access the OCI Console in **nf_node_subnet** as described in the following table:

Table 2-10 Adding Ingress Rules for Node Subnet

Option Name	Description
Source Type	Indicates the source type. Value: CIDR
Source CIDR	Indicates the source of CIDR. Value: <CIDR for Load balancer subnet> **Note**: The value for the variable <CIDR for Node subnet> was determined in a previous step.
IP Protocol	Indicates the internet protocol for which this rule is applicable. Value: TCP
Source Port Range	Keep this option at its default value.
Destination Port Range	Keep this option at its default value.
Description	Keep this option at its default value.

> **Note**
>
> To access ATS API, you can use the above mentioned ingress rule.

Updating Security List for Load Balancer Subnet

1. To update security list for Load balancer subnet, do the following:
   a. Navigate to **nf_lb_subnet**, and then to **nf_lb_security_list**.
   b. Click **Add Ingress Rules**.
   c. Add ingress rule if it is unavailable to access the OCI Console in **nf_lb_subnet** by configuring the options as described in the following table:

Table 2-11 Adding Ingress Rules Load Balancer Subnet

Option Name	Description
Source Type	Indicates the source type. Value: CIDR
Source CIDR	Indicates the source of CIDR. Value: <CIDR for Network which will be used to access ATS GUI> **Note**: Avoid setting its value to 0.0.0.0/0, as it may result in granting access to ATS GUI from all networks, posing a considerable security risk. If you intend for ATS GUI to only be accessible within the Corporate Proxy (VPN), using a Source CIDR value of 0.0.0.0/0 would allow access from outside the Corporate Proxy (VPN) as well.
IP Protocol	Indicates the internet protocol for which this rule is applicable. Value: TCP
Source Port Range	Keep this option at its default value.
Destination Port Range	Indicates the ATS GUI port number. Port 8080 is used for HTTP and 8443 is used for HTTPS. Value: `<ATS service Port>`
Description	Keep this option at its default value.

   d. To access ATS API, create one more ingress rule if it is unavailable by following the previous step with the following changes:

- In the **Source CIDR** option, provide the `<CIDR for Network which will be used to access ATS API>` value.
- In the **Destination Port Range** option, provide the `<ATS Service API Port>` value, which refers to the ATS API port, that is, 5001.

2. To add egress rule if it is unavailable to access the OCI Console in **nf_lb_subnet**, do the following:
   a. In the left navigation pane, click **Egress Rules**.
   b. Click **Add Egress Rules**.
   c. Configure the following options as described in this table:

Table 2-12 Adding Egress Rules Load Balancer Subnet

Option Name	Description
Destination Type	Indicates the destination type. Value: CIDR
Destination CIDR	Indicates the destination of CIDR. Value: <CIDR for Node subnet> **Note**: The value for the variable <CIDR for Node subnet> was determined in the previous step.
IP Protocol	Indicates the internet protocol for which this rule is applicable. Value: TCP
Source Port Range	Keep this option at its default value.
Destination Port Range	Keep this option at its default value.
Description	Keep this option at its default value.

> **Note**
>
> To access ATS API, you can use the above mentioned egress rule.

Updating Security List for Tunneling

1. Log in to the OCI Console using your login credentials.
2. On the OCI Console Dashboard, click **Navigation menu**.

Figure 2-81 Navigation menu

3. Click **Networking**, and then click **Virtual cloud networks** on the preview pane.
4. Click the required OKE cluster VCN.
   A list of all subnets in that VCN is displayed.
5. Save the value of the "IPv4 CIDR Block" field for **nf_operator_subnet**.
   This information can be used in the subsequent steps as value of `<CIDR for Operator subnet>`.

Updating Security list of Node Subnet

1. To update security list of Node subnet, do the following:
   a. Navigate to **nf_node_subnet**, and then to **nf_node_security_list**.
   b. Click **Add Ingress Rules**.
   c. Add ingress rule if it is unavailable to access the OCI Console in **nf_node_subnet** by configuring the options as described in the following table:

Table 2-13 Adding Ingress Rules for Node Subnet Tunneling

Option Name	Description
Source Type	Indicates the source type. Value: CIDR
Source CIDR	Indicates the source of CIDR. Value: <CIDR for Operator subnet> **Note**: The value for the variable <CIDR for Operator subnet> was determined in the previous step.
IP Protocol	Indicates the internet protocol for which this rule is applicable. Value: TCP
Source Port Range	Keep this option at its default value.
Destination Port Range	Keep this option at its default value.
Description	Keep this option at its default value.

> **Note**
>
> To access ATS API, you can use the above mentioned ingress rule.

2.24 Support for Transport Layer Security

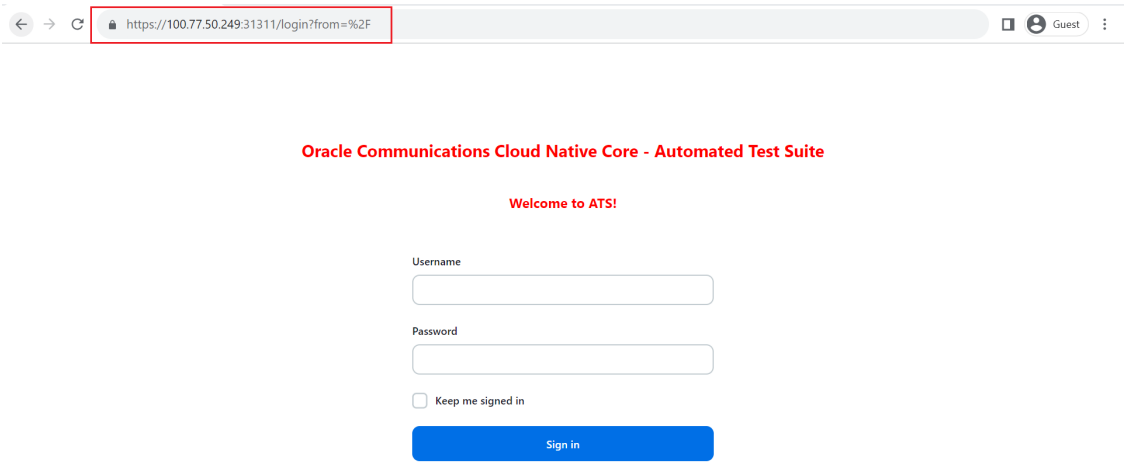
Currently, ATS is accessible through HTTP, which can raise security risks.

With the support of the TLS feature, Jenkins servers have been upgraded to support HTTPS, ensuring a secure and encrypted connection when accessing the ATS dashboard.

To provide encryption, HTTPS uses an encryption protocol known as Transport Layer Security (TLS), which is a widely accepted standard protocol that provides authentication, privacy, and data integrity between two communicating computer applications.

Now, users can access the ATS GUI with the HTTPS protocol instead of the previously used HTTP protocol.

Figure 2-82 Access with TLS



> **Note**
>
> If this feature is not enabled before installation, HTTP will continue to operate.

2.24.1 Deploy ATS with TLS Enabled

Follow the steps in this section to create a Java KeyStore (JKS) file and enable the ATS GUI with HTTPS during installation.

2.24.1.1 Generate JKS File for Jenkins Server

A JKS file needs to be created in order for Jenkins to provide ATS GUI access through HTTPS.

Perform the following steps to generate a JKS file:

Generate the Root Certificate

If a root certificate, for example, `caroot.cert`, is not already available, a user can generate one. Users may use their own files if they have a CA-signed root certificate and key or their own root certificates. The root certificate is used to sign the application, or ATS certificate.

Follow the steps to create and use self-signed certificates:

1. Generate a root key with the following command:

   ```
   openssl genrsa 2048 > <path_to_root_key>
   ```

   For example,

   ```
   openssl genrsa 2048 > caroot.key
   ```

2. Generate a "caroot" certificate with the following command:

```
openssl req -new -x509 -nodes -days 1000 -key
      <path_to_root_key> > <path_to_root_certificate>
```

For example,

```
[cloud-user@star23-bastion-1 cert]$ openssl req -new -x509 -nodes -days
1000 -key caroot.key > caroot.cer
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:KA
Locality Name (eg, city) [Default City]:BLR
Organization Name (eg, company) [Default Company Ltd]:ORACLE
Organizational Unit Name (eg, section) []:CGBU
Common Name (eg, your name or your server's hostname) []:ocats
Email Address []:
[cloud-user@star23-bastion-1 cert]$
```

Generate Application or Client Certificate

Follow the steps to create and edit the `ssl.conf` file:

1. In the **alt_names** section, list the IPs, such as IP.1, IP.2, and so on, that are used to access the ATS GUI:

```
[ req ]
default_bits       = 4096
distinguished_name = req_distinguished_name
req_extensions     = req_ext

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = <Country_Name>
stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = <State_Name>
localityName                = Locality Name (eg, city)
localityName_default        = <Locality_Name>
organizationName            = Organization Name (eg, company)
organizationName_default    = <Org_Name>
commonName                  = Common Name (e.g. server FQDN or YOUR name)
commonName_max              = 64
commonName_default          = <helm_name>.<namespace>.svc.cluster.local

[ req_ext ]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
basicConstraints = critical, CA:FALSE
subjectAltName = critical, @alt_names
```

```
[alt_names]
IP.1 = 127.0.0.1
IP.2 = <IP1>
IP.3 = <IP2>
DNS.1 = <helm_name>.<namespace>.svc.cluster.local
```

For example,

```
[ req ]
default_bits       = 4096
distinguished_name = req_distinguished_name
req_extensions     = req_ext

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = <Country_Name>
stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = <State_Name>
localityName                = Locality Name (eg, city)
localityName_default        = <Locality_Name>
organizationName            = Organization Name (eg, company)
organizationName_default    = <Org_Name>
commonName                  = Common Name (e.g. server FQDN or YOUR name)
commonName_max              = 64
commonName_default          = ocats.scpsvc.svc.cluster.local

[ req_ext ]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
basicConstraints = critical, CA:FALSE
subjectAltName = critical, @alt_names

[alt_names]
IP.1 = 127.0.0.1
IP.2 = 10.75.217.5
IP.3 = 10.75.217.76
DNS.1 = localhost
DNS.2 = ocats.scpsvc.svc.cluster.local
```

Note

- To access the GUI with DNS, make sure that the commonName_default is the same as the DNS name being used.
  - The `/etc/hosts` file in the local system should also be updated with the assigned IP and the DNS mentioned above to open the ATS with DNS. Otherwise, it is not required.
- Ensure the DNS is in this format: <service_name>.<namespace>.svc.cluster.local
- Multiple DNSs, such as DNS.1, DNS.2, and so on, can be added.
- To support the ATS API, add the IP 127.0.0.1 to the list of IPs.

2. Create a Certificate Signing Request (CSR) with the following command:

```
openssl req -config ssl.conf -newkey rsa:2048 -days 1000 -nodes -keyout
<path_to_application_certificate_key> >
<path_to_certificate_signing_request>
```

For example,

```
[cloud-user@star23-bastion-1 cert]$ openssl req -config ssl.conf -newkey
rsa:2048 -days 1000 -nodes -keyout rsa_private_key_pkcs1.key >
ssl_rsa_certificate.csr
Ignoring -days; not generating a certificate
Generating a RSA private key
...+++++
........+++++
writing new private key to 'rsa_private_key_pkcs1.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IN]:
State or Province Name (full name) [KA]:
Locality Name (eg, city) [BLR]:
Organization Name (eg, company) [ORACLE]:
Common Name (e.g. server FQDN or YOUR name) [ocats]:
[cloud-user@star23-bastion-1 cert]$
```

3. Use the command to display the components of the file and verify the configurations:

```
openssl req -text -noout -verify -in
    ssl_rsa_certificate.csr
```

4. Sign in to this CSR file with the root certificate:

```
openssl x509 -extfile ssl.conf -extensions req_ext -req -in
<path_to_certificate_signing_request> -days 1000 -CA
<path_to_root_certificate> -CAkey <path_to_root_key> -set_serial 04 >
<path_to_application_certificate>
```

For example,

```
[cloud-user@star23-bastion-1 cert]$ openssl x509 -extfile ssl.conf -
extensions req_ext -req -in ssl_rsa_certificate.csr -days 1000 -CA ../
caroot.cer -CAkey ../caroot.key -set_serial 04 > ssl_rsa_certificate.crt
Signature ok
subject=C = IN, ST = KA, L = BLR, O = ORACLE, CN = ocats
Getting CA Private Key
[cloud-user@star23-bastion-1 cert]$
```

5. Use the command to verify that the certificate is properly signed by the root certificate:

   ```
   openssl verify -CAfile <path_to_root_certificate>
         <path_to_application_certificate>
   ```

   For example,

   ```
   [cloud-user@star23-bastion-1 cert]$ openssl verify -CAfile caroot.cer
   ssl_rsa_certificate.crt
   ssl_rsa_certificate.crt: OK
   ```

6. Save the generated application certificates and the root certificates.
7. Add the `caroot.cer` to the browser as a trusted author. For more information, see Enable ATS GUI with HTTPS
8. Generate the `.p12 keystore` file with the following command:

   ```
   openssl pkcs12 -inkey <path_to_application_key> -in
   <path_to_application_certificate> -export -out <path_to_p12_certificate>
   ```

   For example,

   ```
   [cloud-user@star23-bastion-1 ocats]$ openssl pkcs12 -inkey
   rsa_private_key_pkcs1.key -in ssl_rsa_certificate.crt -export -out
   certificate.p12
   Enter Export Password:
   Verifying - Enter Export Password:
   ```

9. In the prompt, create a password and save it for future use.
10. Convert the `.p12 keystore` file into a JKS format file:

    ```
    keytool -importkeystore -srckeystore <path_to_p12_certificate> -
    srcstoretype pkcs12 -destkeystore <path_to_jks_file> -deststoretype JKS
    ```

    For example,

    ```
    [cloud-user@star23-bastion-1 cert]$ keytool -importkeystore -srckeystore ./
    certificate.p12 -srcstoretype pkcs12 -destkeystore jenkinsserver.jks -
    deststoretype JKS
    Importing keystore ./certificate.p12 to jenkinsserver.jks...
    Enter destination keystore password:
    Re-enter new password:
    Enter source keystore password:
    Entry for alias 1 successfully imported.
    Import command completed:  1 entries successfully imported, 0 entries
    failed or cancelled
    ```

11. In the prompt, use the same password used while creating `.p12 keystore` file.

> **Note**
>
> Verify that the passwords linked to the `.p12 keystore` and JKS files are the same.

12. The generated file, `jenkinserver.jks`, is provided to the Jenkins server.

2.24.1.2 Enable ATS GUI with HTTPS

Follow the steps to secure or enable TLS on the server.

1. Create a Kubernetes secret by adding the above created files:

```
kubectl create secret generic <secret_name> --from-file=<path_to_jks_file>
--from-file=<path_to_application_certificate> --from-
file=<path_to_application_key> --from-file=<path_to_root_certificate> -n
<namespace>
```

For example,

```
kubectl create secret generic ocats-tls-secret --from-
file=jenkinsserver.jks --from-file=ssl_rsa_certificate.crt --from-
file=rsa_private_key_pkcs1.key --from-file=caroot.cer -n scpsvc
```

Where,

jenkinsserver.jks: This file is needed when `atsGuiTLSEnabled` is set to true. This is necessary to open ATS GUI with secured TLS protocol.

ssl_rsa_certificate.crt: This is client application certificate.

rsa_private_key_pkcs1.key: This is RSA private key.

caroot.cer: This file used during creation of jks file needs to be passed for Jenkins/ATS API communication.

The sample of created secret:

```
[cloud-user@star23-bastion-1 cert]$ kubectl describe secret ocats-tls-
secret -n scpsvc
Name:         ocats-tls-secret
Namespace:    scpsvc
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
caroot.cer:                      1147 bytes
ssl_rsa_certificate.crt:         1424 bytes
jenkinsserver.jks:               2357 bytes
rsa_private_key_pkcs1.key:       1675 bytes
```

2. Apply the following changes in `values.yaml` file.

```
certificates:
  cert_secret_name: "ocats-tls-secret"
  ca_cert: "caroot.cer"
  client_cert: "ssl_rsa_certificate.crt"
  private_key: "rsa_private_key_pkcs1.pem"
  jks_file: "jenkinsserver.jks"
  jks_password: "123456"  #This is the password given to the jks file
while creation.
```

   The user can install the ATS, using the helm install command. Change the `atsGuiTLSEnabled` Helm parameter value to true for ATS to get the certificates and support HTTPS for GUI.

3. A user can now start ATS with HTTPS the protocol.
   The link to open the ATS GUI format is `https://<IP>:<port>`, for example, `https://10.75.217.25:30301`

   The lock symbol in the browser indicates that the server is secured or TLS enabled.

Adding a Certificate in Browser

The created root certificate is added to the browser, either Mozilla Firefox or Chrome.

> **Note**
>
> Future versions of these browsers may involve different menu options. For more information on importing root certificate, see the browser documentation to add a self-signed certificate to the browser as a trusted certificate.

Adding in Windows laptops

1. In the Chrome browser, navigate to the settings and search for security.
2. Click the **security** option that appears next to search.
3. Click the **Manage Device Certificate** option. The Certificate window opens.
4. Click the Trusted root certification authorities bar.
5. Import the `caroot` certificate.
6. Save and restart the browser.

Adding in Mac laptop

1. In the Chrome browser, navigate to the settings and search for security.
2. Click the **security** option that appears next to search.
3. Click the **Manage Device Certificate** option. The Keychain Access window opens.
4. Search the tab certificate and drag and drop the downloaded caroot certificate.
5. Find the uploaded certificate in the list, usually listed by a temporary name.
6. Double click the certificate and expand the Trust option.
7. In **When using this certificate** option, assign it to "always trust".

8. Close the window and validate if it asks for the password.
9. Save and restart the browser.

Adding for Windows and Mac laptop

1. In the Mozilla Firefox browser, navigate to the settings and search for certificates.
2. Click the **View Certificate** that appears next to search. This opens a Certificate Manager window.
3. Navigate to the Authorities section, click the **Import** button, and upload the `caroot` certificate.
4. Click the **Trust** options in the pop-up window and click OK.
5. Save and restart the browser.

2.25 Test Results Analyzer

The Test Results Analyzer is a plugin available in ATS to view pipeline test results based on XML reports. It provides the test results report in a graphical format, which includes consolidated and detailed stack trace results in case of any failures. It allows you to navigate to each and every test.

The test result report shows any one of the following statuses for each test case:

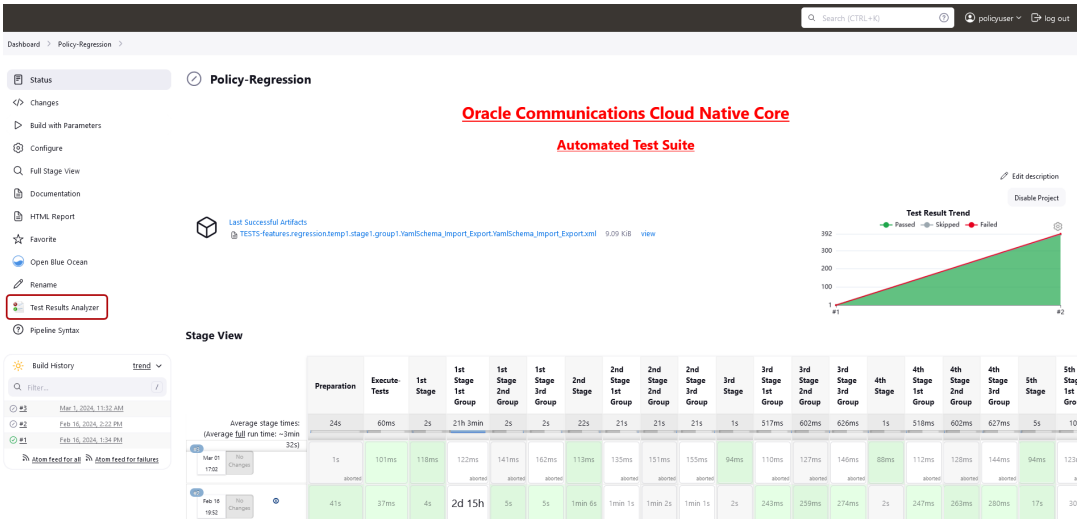
- **PASSED:** If the test case passes.
- **FAILED:** If the test case fails.
- **SKIPPED:** If the test case is skipped.
- **N/A:** If the test cases are not executed in the current build.

2.25.1 Accessing Test Results Analyzer Feature

To access the test results analyzer feature:

1. From the NF home page, click any new feature pipeline or regression pipeline where you want to run this plugin.
2. In the left navigation pane, click **Test Results Analyzer**.

Figure 2-83 Test Results Analyzer Option



When the build completes, the test result report appears. A sample test result report is shown below:

Figure 2-84 Sample Test Result Report

Chart	Package/Class/Testmethod	Passed	Transitions	5	4	3	2	1
☐	features.cust_newfeatures.temp1.Binding_Service.Binding_Service	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Bulk_Import_Export.Bulk_Import_Export	100% (100%)	0	PASSED	PASSED	SKIPPED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.Diameter_GW_Error_Handling.Diameter_GW_Error_Handling	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Dynamic_Policy_Override.Dynamic_Policy_Override.This feature aims to dynamically override the parameters of pcc and session rules during SM Policy Association	100% (100%)	0	PASSED	PASSED	SKIPPED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.IMS_Emergency_Call.IMS_Emergency_Call	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Operators.Operators	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.PolicyActionForBulkPccRuleRemoval.Policy_Action_Bulk_Pccrule_Removal	100% (100%)	0	PASSED	PASSED	SKIPPED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.Policy_Conditions_And_Actions.Policy_Conditions_And_Actions	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Policy_Table.Policy_Table	100% (100%)	0	PASSED	PASSED	PASSED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.Request_AVP.Request_AVP	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Sanity.Sanity	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.SessionRetry.SessionRetry.To verify that SM or User-service retry to alternate FQDN if the	0%	0	FAILED	FAILED	SKIPPED	FAILED	N/A

3. Click any one of the statuses (PASSED, FAILED, SKIPPED) to view the respective feature detail status report.

> **Note**
>
> For N/A status, a detailed status report is not available.

Figure 2-85 Test Result

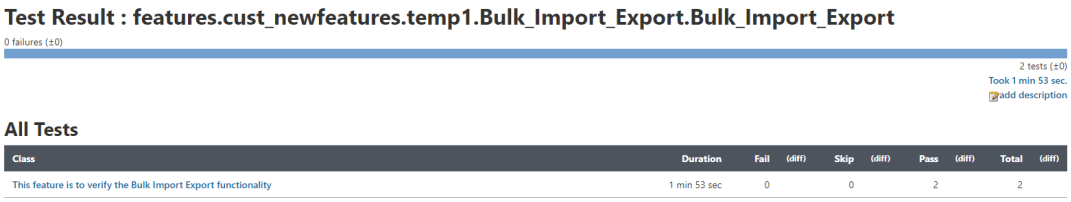
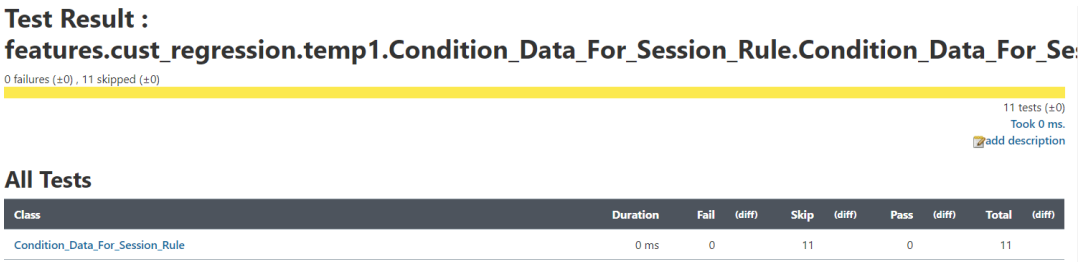


Figure 2-86 Test Result



4. In the case of a rerun, the test cases that passed in the initial run but were skipped in the rerun are considered "passed" in the Test Results Analyzer Report. The following screenshot depicts the scenario: "Variant2_equal_smPolicySnssaiData,Variant2_exist_smPolicyData,Variant2_exist_smPolicyDnnData_dnn" where the test cases passed in the initial run but skipped in the rerun are considered "passed" in general.

Figure 2-87 Test Results

5. Click **PASSED**. The following highlighted message means the test case was passed in the main run but skipped in the rerun.

Figure 2-88 Test Result Info

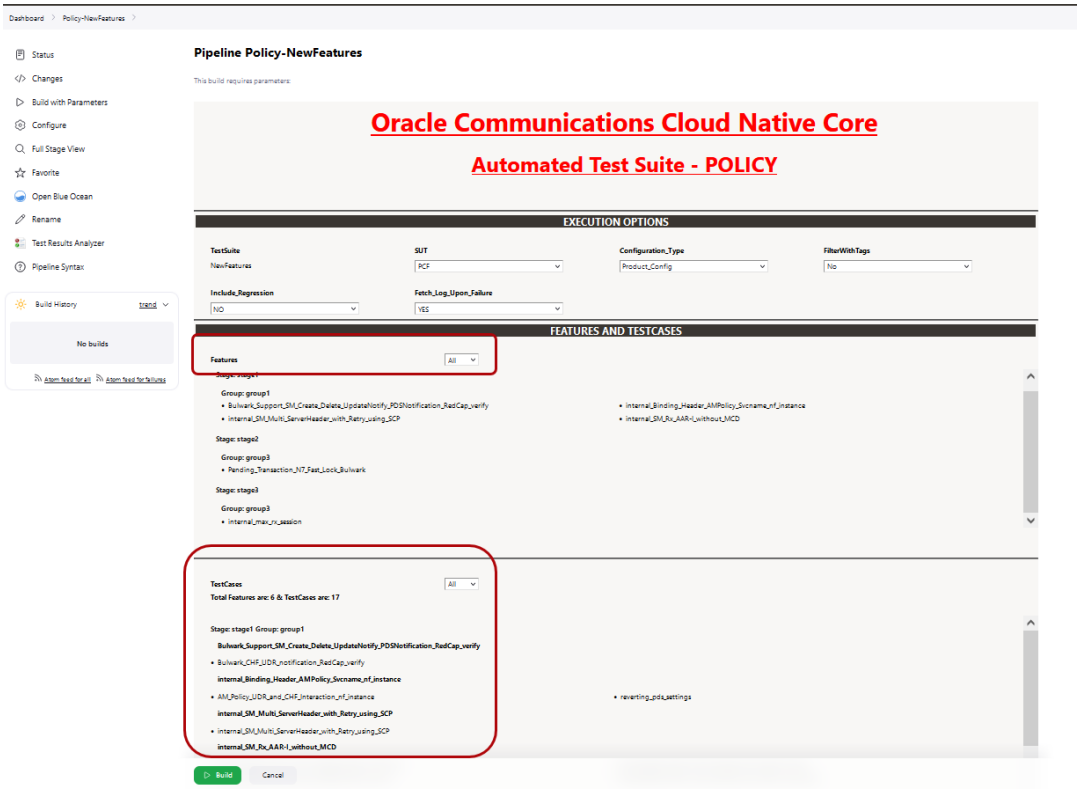
2.26 Support for Test Case Mapping and Count

The Test Case Mapping and Count feature displays the total number of features, test cases, or scenarios and their mapping to each feature in the ATS GUI.

2.26.1 Access Test Case Mapping and Count Feature

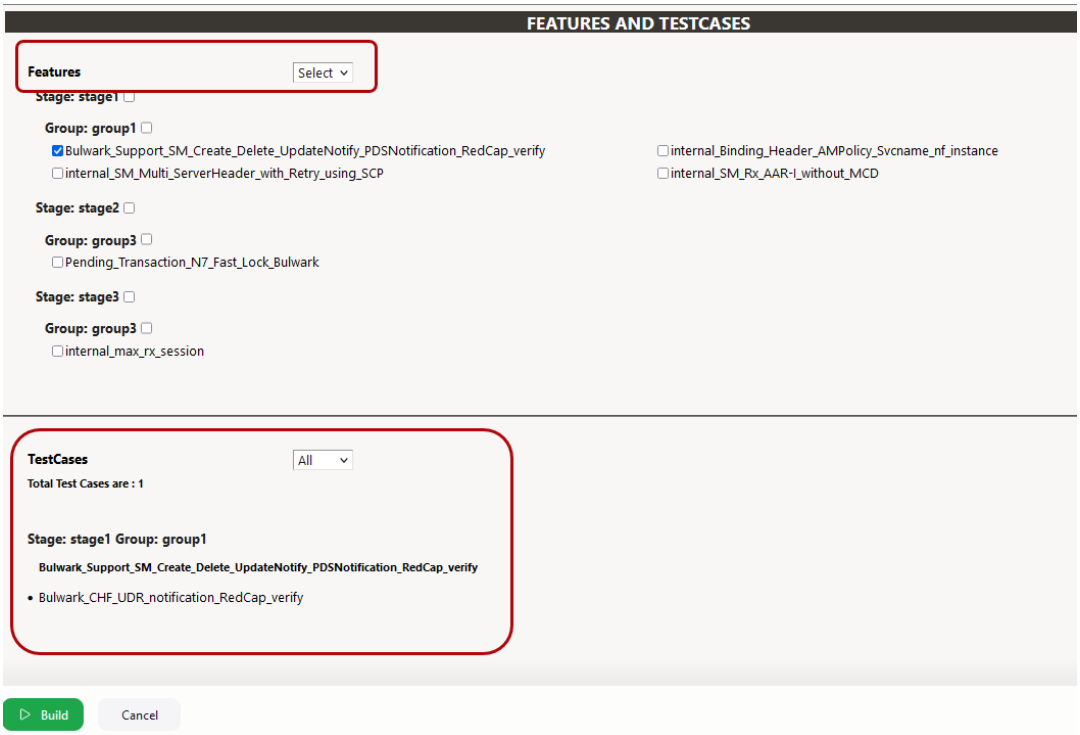
1. On the NF home page, click any new feature or regression pipeline where you want to use this feature.
2. In the left navigation pane, click **Build with Parameters**.
3. On the FEATURE AND TESTCASES section, click **All** from the Features drop-down menu to view the TestCases details mapped to each feature.

Figure 2-89 Test Case Mapping



4. Click **Select** from the **Features** drop-down menu to view the the test cases details.

Figure 2-90 Test Cases Details When Select a Single/MultipleFeatures



3
Installing ATS for Different Network Functions

This section describes how to install ATS for different network functions. It includes:

- Installing ATS for BSF
- Installing ATS for NSSF

3.1 Installing ATS for BSF

The BSF ATS installation procedure covers two steps:

1. Locating and downloading the ATS package for BSF.
2. Deploying ATS and stub pods in Kubernetes cluster.

This includes installation of three stubs (nf1stub, nf11stub, and nf12stub), ocdns-bind stub, and BSF ATS in BSF namespace.

3.1.1 Resource Requirements

This section describes the ATS resource requirements for Binding Support Function.

Overview - Total Number of Resources

The following table describes the overall resource usage in terms of CPUs, memory, and storage:

Table 3-1 BSF - Total Number of Resources

Resource Name	Non-ASM CPU	Non-ASM Memory (GB)	ASM CPU	ASM Memory (GB)
BSF Total	41	36	73	52
ATS Total	11	11	23	17
cnDBTier Total	107.1	175.2	137.1	190.2
Grand Total BSF ATS	159.1	222.2	233.1	259.2

BSF Pods Resource Requirements Details

This section describes the resource requirements, which are needed to deploy BSF ATS successfully.

Table 3-2 BSF Pods Resource Requirements Details

BSF Microservices	Max CPU	Memory (GB)	Max Replica	Isito ASM CPU	Isito ASM Memory (GB)	Non-ASM Total CPU	Non-ASM Memory (GB)	ASM Total CPU	ASM Total Memory (GB)
oc-app-info	1	1	1	2	1	1	1	3	2
oc-diam-gateway	4	2	1	2	1	4	2	6	3
alternate-route	2	4	1	2	1	2	4	4	5
oc-config-server	4	2	1	2	1	4	2	6	3
ocegress_gateway	4	6	1	2	1	4	6	6	7
ocingress_gateway	4	6	1	2	1	4	6	6	7
nrf-client-mngt	1	1	2	2	1	2	2	6	4
oc-audit	2	1	1	2	1	2	1	4	2
oc-config-mgmt	4	2	2	2	1	8	4	12	6
oc-query	2	1	2	2	1	4	2	8	4
oc-perf-info	1	1	2	2	1	2	2	6	4
bsf-management-service	4	4	1	2	1	4	4	6	5
BSF Totals						41	36	73	52

ATS Resource Requirements details for BSF

This section describes the ATS resource requirements, which are needed to deploy BSF ATS successfully.

Table 3-3 ATS Resource Requirements Details

ATS Microservices	Max CPU	Max Memory (GB)	Max Replica	Isito ASM CPU	Isito ASM Memory (GB)	Non-ASM Total CPU	Non-ASM Total Memory (GB)	ASM Total CPU	ASM Total Memory (GB)
ocstub1-py	2	2	1	2	1	2	2	4	3
ocstub2-py	2	2	1	2	1	2	2	4	3

Table 3-3 (Cont.) ATS Resource Requirements Details

ATS Microservices	Max CPU	Max Memory (GB)	Max Replica	Isito ASM CPU	Isito ASM Memory (GB)	Non-ASM Total CPU	Non-ASM Total Memory (GB)	ASM Total CPU	ASM Total Memory (GB)
ocstub3-py	2	2	1	2	1	2	2	4	3
ocats-bsf	3	3	1	2	1	3	3	5	4
ocdns-bind	1	1	1	2	1	1	1	3	2
ocdiam-sim	1	1	1	2	1	1	1	3	2
ATS Totals						11	11	23	17

cnDBTier Resource Requirements Details for BSF ATS

This section describes the cnDBTier resource requirements, which are needed to deploy BSF ATS successfully.

> **Note**
>
> For cnDBTier pods, a minimum of 4 worker nodes are required.

Table 3-4 cnDBTier Resource Requirements Details

cnDBTier Microservices	Min CPU	Min Memory (GB)	Min Replica	Isito ASM CPU	Isito ASM Memory (GB)	Total CPU	Total Memory (GB)	ASM Total CPU	ASM Total Memory (GB)
db_monitor_svc	1	1	1	2	1	1	1	3	2
db_replication_svc	2	12	1	2	1	2	12	4	13
db_backup_manager_svc	0.1	0.2	1	2	1	0.1	0.2	2.1	1.2
ndbappmysqld	8	10	4	2	1	32	40	40	44
ndbmgmd	4	10	2	2	1	8	20	12	22
ndbmtd	10	18	4	2	1	40	72	48	76
ndbmysqld	8	10	2	2	1	16	20	20	22
db_infra_moditor_svc	8	10	1	2	1	8	10	8	10
cnDBTier Total						107.1	175.2	137.1	190.2

3.1.2 Downloading the ATS Package

This section provides information on how to locate and download BSF ATS package file from My Oracle Support (MOS).

Locating and Downloading BSF ATS Package

To locate and download the ATS Image from MOS, perform the following steps:

1. Log in to My Oracle Support using the appropriate credentials.
2. Select the **Patches and Updates** tab.
3. In the **Patch Search** window, click **Product or Family (Advanced)**.
4. Enter *Oracle Communications Cloud Native Core - 5G* in the **Product** field.
5. Select *Oracle Communications Cloud Native Binding Support Function <release_number>* from **Release** drop-down.
6. Click **Search**. The **Patch Advanced Search Results** list appears.
7. Select the required patch from the search results. The Patch Details window appears.
8. Click **Download**. The File Download window appears.
9. Click the **<p********_<release_number>_Tekelec>.zip** file to downlaod the BSF ATS package file.
10. Untar the gzip file `ocats-bsf-tools-25.2.200.0.0.tgz` to access the following files:

    ```
    ocats-bsf-pkg-25.2.200.0.0.tgz
    ocdns-pkg-25.2.204.tgz
    ocstub-pkg-25.2.202.tgz
    ocdiam-sim-25.2.204.tgz
    ```

    The contents included in each of these files are as follow:

    ```
    ocats-bsf-tools-25.2.200.0.0.tgz
    |
    |_ _ _ocats-bsf-pkg-25.2.200.tgz
    | |_ _ _ _ _ _ ocats-bsf-25.2.200.tgz (Helm Charts)
    | |_ _ _ _ _ _ ocats-bsf-images-25.2.200.tar (Docker Images)
    | |_ _ _ _ _ _ ocats-bsf-data-25.2.200.tgz (BSF ATS and Jenkins job Data)
    |
    |_ _ _ocstub-pkg-25.2.202.0.0.tgz
    | |_ _ _ _ _ _ ocstub-py-25.2.202.tgz(Helm Charts)
    | |_ _ _ _ _ _ ocstub-py-image-25.2.202.tar (Docker Images)
    |
    |_ _ _ocdns-pkg-25.2.204.0.0.tgz
    | |_ _ _ _ _ _ ocdns-bind-25.2.204.tgz(Helm Charts)
    | |_ _ _ _ _ _ ocdns-bind-image-25.2.204.tar (Docker Images)
    |
    |_ _ _ocdiam-pkg-25.2.204.0.0.tgz
    |      |_ _ _ _ _ _ ocdiam-sim-25.2.204.tgz(Helm Charts)
    |      |_ _ _ _ _ _ ocdiam-sim-image-25.2.204.tar (Docker Images)
    ```

11. Copy the tar file from the downloaded package to CNE, OCI, or Kubernetes cluster where you want to deploy ATS.

3.1.3 Deploy ATS with TLS Enabled

> **Note**
>
> - OCATS and Python stubs support both TLS 1.2. and TLS 1.3.
> - DiamSim pod do not support secure calls.

Follow the steps in this section to create a Java KeyStore (JKS) file and enable the BSF ATS GUI with HTTPS during installation.

3.1.3.1 Generate JKS File for Jenkins Server

To access Jenkins ATS GUI access through HTTPS, a JKS file should be created.

Perform the following steps to generate the JKS file:

Generate the Root Certificate

1. If the user has a Certificate Authority (CA) signed root certificate such as `caroot.cert` and key, then the user can use those files.
2. If the root certificate is not already available, the user can generate one self signed root certificate. This root certificate created needs to be added to the `truststore` such as a Browser like Firefox or Chrome. User can follow the Browser specific documentation to upload the root certificate. The root certificate is used to sign the application, or ATS certificate.
3. Generate a root key with the following command:

   ```
   openssl genrsa 2048 > caroot.key
   ```

   This will generate a key called `caroot.key`
4. Generate a `caroot` certificate with the following command:

   ```
   openssl req -new -x509 -nodes -days 1000 -key <root_key> >
   <root_certificate>
   ```

   For example,

   ```
   [cloud-user@platform-bastion-1]$ openssl req -new -x509 -nodes -days 1000 -
   key caroot.key > caroot.cer
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [XX]:IN
   ```

```
State or Province Name (full name) []:KA
Locality Name (eg, city) [Default City]:BLR
Organization Name (eg, company) [Default Company Ltd]:ORACLE
Organizational Unit Name (eg, section) []:CGBU
Common Name (eg, your name or your server's hostname) []:ocats
Email Address []:
[cloud-user@platform-bastion-1]$
```

Generate Application or Client Certificate

1. Create a `ssl.conf` file.
2. Edit the `ssl.conf` file. In the "[alt_names]" section, list the IPs that are used to access ATS GUI as shown in the following sample `ssl.conf` file:

```
[ req ]
default_bits       = 4096
distinguished_name = req_distinguished_name
req_extensions     = req_ext

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = IN
stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = KN
localityName                = Locality Name (eg, city)
localityName_default        = BLR
organizationName            = Organization Name (eg, company)
organizationName_default    = ORACLE
commonName                  = Common Name (e.g. server FQDN or YOUR name)
commonName_max              = 64
commonName_default          = ocats.ocbsf.svc.cluster.local

[ req_ext ]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
basicConstraints = critical, CA:FALSE
subjectAltName = critical, @alt_names

[alt_names]
IP.1 = 127.0.0.1
IP.2 = 10.75.217.5
IP.3 = 10.75.217.76
DNS.1 = localhost
DNS.2 = ocats.ocbsf.svc.cluster.local
```

> **Note**
>
> - To access the GUI with DNS, make sure that the commonName_default is the same as the DNS name being used.
>   - Ensure the DNS is in this format:`<service_name>.<namespace>.svc.cluster.local`
>
>   Multiple DNSs, such as DNS.1, DNS.2, and so on, can be added.
> - To support the ATS API, it is necessary to add the IP 127.0.0.1 to the list of IPs.

3. Create a Certificate Signing Request (CSR) with the following command:

   ```
   openssl req -config ssl.conf -newkey rsa:2048 -days 1000 -nodes -keyout
   rsa_private_key_pkcs1.key > ssl_rsa_certificate.csr
   ```

   Output:

   ```
   [cloud-user@platform-bastion-1 ocbsf]$ openssl req -config ssl.conf -
   newkey rsa:2048 -days 1000 -nodes -keyout rsa_private_key_pkcs1.key >
   ssl_rsa_certificate.csr
   Ignoring -days; not generating a certificate
   Generating a RSA private key
   ...+++++
   ........+++++
   writing new private key to 'rsa_private_key_pkcs1.key'
   -----
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [IN]:
   State or Province Name (full name) [KA]:
   Locality Name (eg, city) [BLR]:
   Organization Name (eg, company) [ORACLE]:
   Common Name (e.g. server FQDN or YOUR name) [ocbsf]:
   [cloud-user@platform-bastion-1 ocbsf]$
   ```

4. To display all the components of the CSR file and to verify the configurations run the following command:

   ```
   openssl req -text -noout -verify -in ssl_rsa_certificate.csr
   ```

5. Sign the CSR file with root certificate by running the following command:

   ```
   openssl x509 -extfile ssl.conf -extensions req_ext -req -
   inssl_rsa_certificate.csr -days 1000 -CA ../caroot.cer -CAkey ../
   caroot.key -set_serial 04 > ssl_rsa_certificate.crt
   ```

Output:

```
[cloud-user@platform-bastion-1 ocbsf]$ openssl x509 -extfile ssl.conf -
extensions req_ext -req -in ssl_rsa_certificate.csr -days 1000 -CA ../
caroot.cer -CAkey ../caroot.key -set_serial 04 > ssl_rsa_certificate.crt
Signature ok
subject=C = IN, ST = KA, L = BLR, O = ORACLE, CN = ocbsf
Getting CA Private Key
[cloud-user@platform-bastion-1 ocbsf]$
```

6. Verify if the certificate is signed by the root certificate by running the following command:

```
[cloud-user@platform-bastion-1 ocbsf]$ openssl verify -CAfile caroot.cer
ssl_rsa_certificate.crt
```

Output:

```
[cloud-user@platform-bastion-1 ocbsf]$ openssl verify -CAfile caroot.cer
ssl_rsa_certificate.crt
ssl_rsa_certificate.crt: OK
```

7. Save the generated application certificate and root certificate.
8. Add the `caroot.cer` to the browser as a trusted author.
9. The generated application/client certificates cannot be directly given to the Jenkins server. Hence generate the `.p12 keystore` file for the client certificate with the following command:

```
[cloud-user@platform-bastion-1 ocbsf]$ openssl pkcs12 -inkey
rsa_private_key_pkcs1.key -inssl_rsa_certificate.crt -export-out
certificate.p12
Enter Export Password:
Verifying - Enter Export Password:
```

10. In the prompt, create a password and save it for future use.
11. Convert the .p12 keystore file into a JKS format file using the following command:

```
[cloud-user@platform-bastion-1 ocbsf]$ keytool -importkeystore -
srckeystore ./certificate.p12 -srcstoretype pkcs12 -destkeystore
jenkinsserver.jks -deststoretype JKS
```

Output:

```
[cloud-user@platform-bastion-1 ocbsf]$ keytool -importkeystore -
srckeystore ./certificate.p12 -srcstoretype pkcs12 -destkeystore
jenkinsserver.jks -deststoretype JKS
Importing keystore ./certificate.p12 to jenkinsserver.jks...
Enter destination keystore password:
```

```
Re-enter new password:
Enter source keystore password:
Entry for alias 1 successfully imported.
Import command completed:  1 entries successfully imported, 0 entries
failed or cancelled
```

12. In the prompt, use the same password used while creating `.p12 keystore` file.

> **Note**
>
> Ensure that the .p12 keystore and JKS files has the same passwords.

13. The generated JKS file, `jenkinserver.jks` is added to the Jenkins path, where Jenkins server can access it.
14. Update following values in ATS values.yaml file

```
certificates:
  cert_secret_name: "ocats-bsf-tls-secret"
  ca_cert: "caroot.cer"
  client_cert: "ssl_rsa_certificate.crt"
  private_key: "rsa_private_key_pkcs1.key"
  jks_file: "jenkinsserver.jks"  # This parameter is needed when
atsGuiTLSEnabled is set to true. This file is necessary for ATS GUI to be
opend with secured TLS protocol. The file caroot.pem, used during creation
of jks file needs to be passed for Jenkins/ATS API communication.
```

For more details about the ATS TLS feature, refer to Deploy ATS with TLS Enabled section.

3.1.3.2 Enable TLS on Python Stubs

Before starting the creation of certificate, user should have the path details of `caroot.cer` file.

1. Create a `ssl.conf` file.
2. Edit `ssl.conf` file. Ensure that the DNS is in the format of `*.<namespace>.svc`
   A sample **stub_ssl.conf** file:

```
[ req ]
default_bits       = 4096
distinguished_name = req_distinguished_name
req_extensions     = req_ext

[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = IN
stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = KN
localityName                = Locality Name (eg, city)
localityName_default        = BLR
organizationName            = Organization Name (eg, company)
organizationName_default    = ORACLE
commonName                  = svc.cluster.local
```

```
[ req_ext ]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
basicConstraints = critical, CA:FALSE
subjectAltName = critical, @alt_names

[alt_names]
IP.1 = 127.0.0.1
DNS.1 = *.ocats.svc
```

3. Create a Certificate Signing Request (CSR) for the stubs using the following command:

```
openssl req -config stub_ssl.conf -newkey  rsa:2048 -days 1000 -nodes -
keyout rsa_private_key_stub_pkcs2.key > stub_ssl_rsa_certificate1.csr
```

Output:

```
[cloud-user@platform-bastion-1 stub_certs]$ openssl req -config
stub_ssl.conf -newkey rsa:2048 -days 1000 -nodes -keyout
rsa_private_key_stub_pkcs2.key > stub_ssl_rsa_certificate1.csr
Ignoring -days; not generating a certificate
Generating a RSA private key
.....................+++++
...+++++
writing new private key to 'rsa_private_key_stub_pkcs2.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IN]:
State or Province Name (full name) [KN]:
Locality Name (eg, city) [BLR]:
Organization Name (eg, company) [ORACLE]:
svc.cluster.local []:*.ocbsf.svc
```

4. Sign the certificate with the CA root using the following command:

```
openssl x509 -extfile stub_ssl.conf -extensions req_ext -req -in
stub_ssl_rsa_certificate1.csr -days 1000 -CA  ../ocbsf-caroot.cer -
CAkey ../ocbsf-caroot.key -set_serial 05 >  stub_ssl_rsa_certificate1.crt
```

Output:

```
[cloud-user@platform-bastion-1 stub_certs]$ openssl x509 -extfile
stub_ssl.conf -extensions req_ext -req -in stub_ssl_rsa_certificate1.csr -
days 1000 -CA ../ocbsf-caroot.cer -CAkey ../ocbsf-caroot.key -set_serial
05 > stub_ssl_rsa_certificate1.crt
Signature ok
```

```
subject=C = IN, ST = KN, L = BLR, O = ORACLE, CN = *.ocbsf.svc
Getting CA Private Key
```

5. Create a secret for the stub and associate it with the namespace using the following command:

   ```
   kubectl create secret generic ocats-stub-secret1 --from-
   file=stub_ssl_rsa_certificate1.crt --from-
   file=rsa_private_key_stub_pkcs2.key --from-file=../ocbsf-caroot.cer -n
   ocbsf
   ```

   Output:

   ```
   [cloud-user@platform-bastion-1 ocbsf]$ kubectl create secret generic ocbsf-
   stub-secret1 --from-file=stub_ssl_rsa_certificate1.crt --from-
   file=rsa_private_key_stub_pkcs2.key --from-file=../ocbsf-caroot.cer -n
   ocbsf
   secret/ocats-stub-secret1 created
   ```

6. Update `values.yaml` file of each Python stub in specific NF namespace with following details:

   ```
   NF: "<NF-Name>"
   cert_secret_name: "ocats-stub-secret"
   ca_cert: "ocbsf-caroot.cer"
   client_cert: "ocbsf-stub_ssl_rsa_certificate.crt"
   private_key: "ocbsf-rsa_private_key_stub_pkcs1.key"
   expose_tls_service: true
   CLIENT_CERT_REQ: true
   ```

   > **Note**
   >
   > If the Helm, `cert_secret_name` parameter, is null, then `ca_cert`, `client_cert`, and `private_key` values are not considered by the TLS.

7. Ensure to update the deployment of all the Python stubs installed on the setup.

For more details about the ATS TLS feature, refer to Support for Transport Layer Security section.

3.1.3.3 Enable TLS on Diam Sim

Before creating the certificate, user should have path details of the `caroot.cer` and `caroot.key` files.

> **Note**
>
> - Use the `caroot.cer` and `caroot.key` files that are created in the Enable TLS on Python Stubs section.
> - Replace all occurrences of NAMESPACE with your actual namespace value.

1. For diam-sim, the following parameters are exposed in values.yaml to configure TLS:

```
tls:
  enabled: false
  secretName: 'dcli-tls-secret'
  rsaPrivateKeyFileName: 'dcli-key.pem'
  ecdsaPrivateKeyFileName: 'dcli-ecdsa-private-key.pem'
  rsaCertificateFileName: 'dcli-cert.crt'
  ecdsaCertifacateFileName: 'dcli-ecdsa-certificate.crt'
  caBundleFileName: 'ca-cert.cer'
```

2. By default, **tls.enabled** is **false.** Set this to **true** to enable TLS on diam-sim.
3. Default **secretName** is **dcli-tls-secret.** . You can change this name as per your requirement. Make sure that name of the secret that is created should match with the name provided for secretName.
   Example Output:

```
tls:
  enabled: true
  secretName: '${NAMESPACE}-dcli-tls-secret'
  rsaPrivateKeyFileName: '${NAMESPACE}-dcli-key.pem'
  ecdsaPrivateKeyFileName: 'dcli-ecdsa-private-key.pem'
  rsaCertificateFileName: '${NAMESPACE}-dcli-cert.crt'
  ecdsaCertifacateFileName: 'dcli-ecdsa-certificate.crt'
  caBundleFileName: 'caroot.cer'
```

4. Certificate and private key generation:

```
openssl genrsa -out ${NAMESPACE}-dcli-key.pem 4096

openssl req -new -sha256 -key ${NAMESPACE}-dcli-key.pem -out ${NAMESPACE}-
dcli-csr.csr -subj "/CN=ocbsf-occnp-diam-gateway.${NAMESPACE}.svc"

openssl x509 -req -sha256 -days 1000 \
    -in ${NAMESPACE}-dcli-csr.csr \
    -CA caroot.cer \
    -CAkey caroot.key \
    -out ${NAMESPACE}-dcli-cert.pem \
    -CAcreateserial

mv ${NAMESPACE}-dcli-cert.pem ${NAMESPACE}-dcli-cert.crt
```

5. Creating kubernetes secrets and helm install:

Keep these files in the defined Kubernetes secrets variable defined in **secretName** parameter of the values.yaml file.

```
kubectl create secret generic ${NAMESPACE}-dcli-tls-secret --from-file=$
{NAMESPACE}-dcli-key.pem --from-file=caroot.cer --from-file=${NAMESPACE}-
dcli-cert.crt -n ${NAMESPACE}
```

```
[cloud-user@platform-bastion-1 Certificates]$ kubectl get secrets -n ocbsf
NAME                                    TYPE                 DATA   AGE
ocbsf-db-pass                           Opaque               2      36h
ocbsf-li-iteration                      Opaque               1      36h
ocbsf-li-iv                             Opaque               1      36h
ocbsf-li-pass                           Opaque               1      36h
ocbsf-li-salt                           Opaque               1      36h
ocbsf-privileged-db-pass                Opaque               2      36h
sh.helm.release.v1.nf11stub.v1          helm.sh/release.v1   1      33h
sh.helm.release.v1.nf12stub.v1          helm.sh/release.v1   1      33h
sh.helm.release.v1.nf1stub.v1           helm.sh/release.v1   1      33h
sh.helm.release.v1.ocats-bsf.v1         helm.sh/release.v1   1      33h
sh.helm.release.v1.ocdiam-sim.v1        helm.sh/release.v1   1      33h
sh.helm.release.v1.ocdns-stub.v1        helm.sh/release.v1   1      33h
sh.helm.release.v1.bsf.v1               helm.sh/release.v1   1      34h
[cloud-user@platform-bastion-1 Certificates]$ kubectl create secret generic
ocbsf-dcli-tls-secret --from-file=ocbsf-dcli-key.pem --from-file=caroot.cer --
from-file=ocbsf-dcli-cert.crt -n ocbsf
secret/dcli-tls-secret created

[cloud-user@platform-bastion-1 Certificates]$ kubectl get secrets -n ocbsf
NAME                                    TYPE                 DATA   AGE
ocbsf-dcli-tls-secret                   Opaque               3      42s
ocbsf-db-pass                           Opaque               2      36h
ocbsf-li-iteration                      Opaque               1      36h
ocbsf-li-iv                             Opaque               1      36h
ocbsf-li-pass                           Opaque               1      36h
ocbsf-li-salt                           Opaque               1      36h
ocbsf-privileged-db-pass                Opaque               2      36h
sh.helm.release.v1.nf11stub.v1          helm.sh/release.v1   1      33h
sh.helm.release.v1.nf12stub.v1          helm.sh/release.v1   1      33h
sh.helm.release.v1.nf1stub.v1           helm.sh/release.v1   1      33h
sh.helm.release.v1.ocats-bsf.v1         helm.sh/release.v1   1      33h
sh.helm.release.v1.ocdiam-sim.v1        helm.sh/release.v1   1      33h
sh.helm.release.v1.ocdns-stub.v1        helm.sh/release.v1   1      33h
sh.helm.release.v1.bsf.v1               helm.sh/release.v1   1      34h
```

3.1.3.4 Enable ATS GUI with HTTPS

Follow the steps to secure or enable TLS on the server.

1. Create a Kubernetes secret by adding the above created files:

```
kubectl create secret generic ocats-tls-secret --from-
file=jenkinsserver.jks --from-file=ssl_rsa_certificate.crt --from-
file=rsa_private_key_pkcs1.key --from-file=caroot.cer -n ocbsf
```

Where,

jenkinsserver.jks: This file is needed when `atsGuiTLSEnabled` is set to true. This is necessary to open ATS GUI with secured TLS protocol.

ssl_rsa_certificate.crt: This is client application certificate.

rsa_private_key_pkcs1.key: This is RSA private key.

caroot.cer: This file used during creation of jks file needs to be passed for Jenkins/ATS API communication.

The sample of created secret:

```
[cloud-user@platform-bastion-1 ~]$ kubectl describe secret ocats-tls-
secret -n ocbsf
Name:         ocats-tls-secret
Namespace:    ocats
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
caroot.cer:                       1147 bytes
ssl_rsa_certificate.crt:          1424 bytes
jenkinsserver.jks:                2357 bytes
rsa_private_key_pkcs1.key:        1675 bytes
```

2. Apply the following changes in `values.yaml` file.

```
certificates:
  cert_secret_name: "ocats-tls-secret"
  ca_cert: "caroot.cer"
  client_cert: "ssl_rsa_certificate.crt"
  private_key: "rsa_private_key_pkcs1.pem"
  jks_file: "jenkinsserver.jks" # This parameter is needed when
atsGuiTLSEnabled
      is set to true. This file is necessary for ATS GUI to  be opend with
secured TLS protocol.
  jks_password: "123456"  #This is the password given to the jks file
while creation.
```

The user can install the ATS, using the helm install command. Change the `atsGuiTLSEnabled` Helm parameter value to true for ATS to get the certificates and support HTTPS for GUI.

3. Upload the `caroot.cer` file to the browser, before accessing it using `https` protocol.
For more details about the uploading the file to the browser, refer *Adding a Certificate in Browser* section in Enable ATS GUI with HTTPS.

4. A user can now start ATS with HTTPS the protocol. The link to open the ATS GUI format is `https://<IP>:<port>`, for example, `https://10.75.217.25:30301`.
The lock symbol in the browser indicates that the server is secured or TLS enabled.

3.1.4 Deploy BSF ATS with Authenticated Prometheus communication

Using the Authentication Prometheus communication feature, ATS can communicate with Prometheus, whether or not authentication is enabled, supporting both secure and non-secure setups. Following values can be configured to support authenticated Prometheus communication in the *values.yaml* file:

```
#The below section is related to the deployments where prometheus needs
authenticated token to request for metrics/alerts
PrometheusAuthEnabled: false  #Set this to true only when prometheus requests
require a token in the header for authentication
PrometheusTLSEnabled: false   #Set this to true only when prometheus is a
https deployment.
#PrometheusScrapeInterval: "60" #Uncomment this line only when scrapeInterval
is any other value than 60 seconds.(ATS will assume 60s as default if no
response comes from prometheus config.)
```

For more information about the above Helm parameters, see Table 2-5.

When **PrometheusAuthEnabled** is set to **true**:

- Ensure that ClusterRole and ClusterRoleBinding are created with the required permissions.
- Ensure that Prometheus Server IP and Prometheus Server Port are set accordingly. For more information about setting the Prometheus Server IP and Prometheus Server Port, see the Running BSF_NewFeatures Pipeline section.

> **Note**
>
> - While communicating with Prometheus, ATS currently does not support TLS (certificate-based authentication).
> - If Prometheus is deployed over HTTPS, the **PrometheusTLSEnabled** parameter needs to be set to ***true*** which allows ATS to connect to Prometheus using either HTTP or HTTPS as needed. Even when connecting through HTTPS, the connection remains insecure because ATS does not manage any certificates for this communication.

3.1.5 Pushing the Images to Customer Docker Registry

This section describes the pre-deployment steps for deploying ATS and stub pods.

Preparing to deploy ATS and Stub Pods in Kubernetes Cluster

To deploy ATS and Stub pods in a Kubernetes Cluster, perform the following steps:

1. Run the following command to extract the tar file content:
   ```
   tar -zxvf ocats-bsf-tools-25.2.200.0.0.tgz
   ```

The output of this command is:

```
ocats-bsf-pkg-25.2.200.tgz
ocstub-pkg-25.2.202.tgz
ocdns-pkg-25.2.204.tgz
ocdiam-pkg-25.2.204.0.0tgz
```

2. Go to the `ocats-bsf-tools-25.2.200.0.0` folder and run the following command to extract the helm charts and docker images of ATS:
   `tar -zxvf ocats-bsf-pkg-25.2.200.0.0.tgz`

   The output of this command is:

   ```
   ocats-bsf-25.2.200.tgz
   ocats-bsf-images-25.2.200.tar
   ocats-bsf-data-25.2.200.tgz
   ```

3. Run the following command in your cluster to load the ATS docker image:
   `docker load --input ocats-bsf-images-25.2.200.tar`
4. Run the following commands to tag and push the ATS images

   ```
   docker tag ocats-bsf:25.2.200 <registry>/ocats-bsf:25.2.200
   docker push <registry>/ocats-bsf:25.2.200
   ```

   Example:

   ```
   docker tag ocats-bsf:25.2.200 localhost:5000/ocats-bsf:25.2.200
   docker push localhost:5000/ocats-bsf:25.2.200
   ```

5. Run the following command to untar the helm charts, in `ocats-bsf-25.2.200.tgz`

   ```
   tar -zxvf ocats-bsf-25.2.200.tgz
   ```

6. Update the registry name, image name and tag in the `ocats-bsf/values.yaml` file as required. For this, you need to update the `image.repository` and `image.tag` parameters in the `ocats-bsf/values.yaml` file.
7. In the `ocats-bsf/values.yaml` file, the `atsFeatures` parameter is configured to control ATS feature deliveries.

   ```
   atsFeatures:  ## DO NOT UPDATE this section without My Oracle Support
   team's support
     testCaseMapping: true              # To display Test cases on GUI along
   with Features
     logging: true                      # To enable feature to collect
   applogs in case of failure
     lightWeightPerformance: false      # The Feature is not implemented yet
     executionWithTagging: true         # To enable Feature/Scenario
   execution with Tag
     scenarioSelection: false           # The Feature is not implemented yet
     parallelTestCaseExecution: true    # To run ATS features parallel
     parallelFrameworkChangesIntegrated: true # To run ATS features parallel
   ```

```
  mergedExecution: false              # To execute ATS Regression and
NewFeatures pipelines together in merged manner
  individualStageGroupSelection: false  # The Feature is not implemented
yet
  parameterization: true              # When set to false, the
Configuration_Type parameter on the GUI will not be available.
  atsApi: true                        # To trigger ATS using ATS API
  healthcheck: true                   # TO enable/disable ATS Health Check.
  atsGuiTLSEnabled: false             # To run ATS GUI in https mode.
  atsCommunicationTLSEnabled: false  #If set to true, ATS will get
necessary variables to communicate with SUT, Stub or other NFs with TLS
enabled. It is not required in ASM environment.
```

> **Note**
>
> It is recommended to avoid altering `atsFeatures` flags.

3.1.6 Configuring ATS

3.1.6.1 Enabling Static Port

To enable static port, in the `ocats-bsf/values.yaml` file under the service section, set the value of `staticNodePortEnabled` parameter to `true` and enter a valid nodePort value for `staticNodePort` parameter.

```
service:
  customExtension:
    labels: {}
    annotations: {}
  type: LoadBalancer
  ports:
    http:
      port: "8080"
      staticNodePortEnabled: false
      staticNodePort: ""
```

3.1.6.2 Enable Static API Node Port

To enable static API node port, in the ocats-bsf/values.yaml file under the **service** section, set the value of **staticNodePortEnabled** parameter to true and enter a valid nodePort value for **staticNodePort** parameter. The following is a snippet of the service section in the yaml file:

```
service:
  customExtension:
    labels: {}
    annotations: {}
  type: LoadBalancer
  ports:
    api:
      port: "5001"
```

```
    staticNodePortEnabled: false
    staticNodePort: ""
```

3.1.6.3 Service Account Requirements

To run BSF-ATS, use the following rules to create a service account:

```
rules:
- apiGroups: ["extensions"]
  resources: ["deployments", "replicasets"]
  verbs: ["watch", "get", "list", "update"]
- apiGroups: ["apps"]
  resources: ["deployments", "replicasets"]
  verbs: ["watch", "get", "list", "update"]
- apiGroups: [""]
  resources: ["pods", "services", "secrets", "configmaps"]
  verbs: ["watch", "get", "list", "delete", "update", "create"]
- apiGroups: [""]
  resources: ["pods/log"]
  verbs: ["get", "list"]
```

3.1.6.4 Enabling Aspen Service Mesh

This section provides information on how to enable Aspen service mesh while deploying ATS for Binding Support Function. The configurations mentioned in this section are optional and should be performed only if ASM is required.

To enable service mesh for BSF ATS, perform the following steps:

1. In the **service** section of the `values.yaml` file, the **serviceMeshCheck** parameter is set to `false` by default. To enable service mesh, set the value for **serviceMeshCheck** to `true`. The following is a snippet of the service section in the yaml file:

```
service:
  customExtension:
    labels: {}
    annotations: {}
  type: LoadBalancer
  ports:
    https:
      port: "8443"
      staticNodePortEnabled: false
      staticNodePort: ""
    http:
      port: "8080"
      staticNodePortEnabled: false
      staticNodePort: ""
    api:
      port: "5001"
      staticNodePortEnabled: false
      staticNodePort: ""
  serviceMeshCheck: true
```

2. If the ASM is not enabled on the global level for the namespace, run the following command to enable it before deploying the ATS:

   ```
   kubectl label --overwrite namespace <namespace_name> istio-
   injection=enabled
   ```

   For example:

   ```
   kubectl label --overwrite namespace ocbsf istio-injection=enabled
   ```

3. Uncomment and add the following annotation under the **lbDeployments** and **nonlbDeployments** section of the global section in `values.yaml` file as follows:
   `traffic.sidecar.istio.io/excludeInboundPorts: "9000"`

   `traffic.sidecar.istio.io/excludeOutboundPorts: "9000"`

   The following is a snippet from the `values.yaml` of BSF:

   ```
   /home/cloud-user/ocats-bsf/ocats-bsf-tools-25.2.200.0.0/ocats-bsf-
   pkg-25.2.200.0.0/ocats-bsf/
   vim values.yaml

    customExtension:
       allResources:
         labels: {}
         annotations: {
         #Enable this section for service-mesh based installation
            traffic.sidecar.istio.io/excludeInboundPorts: "9000",
            traffic.sidecar.istio.io/excludeOutboundPorts: "9000"
           }


   lbDeployments:
         labels: {}
         annotations: {
           traffic.sidecar.istio.io/excludeInboundPorts: "9000",
           traffic.sidecar.istio.io/excludeOutboundPorts: "9000"}
   ```

4. If service mesh is enabled, then create a destination rule for fetching the metrics from the Prometheus. In most of the deployments, Prometheus is kept outside the service mesh so you need a destination rule to communicate between TLS enabled entity (ATS) and non-TLS entity (Prometheus). You can create a destination rule using the following sample yaml file:

   ```
   kubectl apply -f - <<EOF

   apiVersion: networking.istio.io/v1alpha3
   kind: DestinationRule
   metadata:
     name: prometheus-dr
     namespace: ocats
   spec:
     host: oso-prometheus-server.pcf.svc.cluster.local
     trafficPolicy:
       tls:
   ```

```
      mode: DISABLE
EOF
```

In the destination rule:

- **name** indicates the name of destination rule.
- **namespace** indicates where the ATS is deployed.
- **host** indicates the hostname of the prometheus server.

5. Update the `ocbsf_custom_values_servicemesh_config_25.2.200.yaml` with the below additional configuration under virtualService section for Egress Gateway:

```
virtualService:
  - name: nrfvirtual1
    host: ocbsf-ocbsf-egress-gateway
    destinationhost: ocbsf-ocbsf-egress-gateway
    port: 8000
    exportTo: |-
      [ "." ]
    attempts: "0"
```

Where,

host or destination name uses the format - <release_name>-<egress_svc_name>.

You must update the host or destination name as per the deployment.

6. For ServerHeader and SessionRetry features, the user needs to perform the following configurations under the envoyFilters for nf1stub, nf11stub, and nf12stub in the `ocbsf-servicemesh-config-custom-values-25.2.200.yaml`:

> **Note**
>
> occnp_custom_values_servicemesh_config yaml file and helm charts version names would differ based on the deployed BSF NF version. For example, "occnp_custom_values_servicemesh_config_24.3.0.yaml" or "occnp_custom_values_servicemesh_config_24.3.1.yaml".

```
envoyFilters:
  - name: serverheaderfilter-nf1stub
    labelselector: "app: nf1stub-ocstub-py"
    configpatch:
      - applyTo: NETWORK_FILTER
        filtername: envoy.filters.network.http_connection_manager
        operation: MERGE
        typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionM
anager
        configkey: server_header_transformation
        configvalue: PASS_THROUGH
  - name: serverheaderfilter-nf11stub
    labelselector: "app: nf11stub-ocstub-py"
    configpatch:
      - applyTo: NETWORK_FILTER
```

```
        filtername: envoy.filters.network.http_connection_manager
        operation: MERGE
        typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionM
anager
        configkey: server_header_transformation
        configvalue: PASS_THROUGH
  - name: serverheaderfilter-nf12stub
    labelselector: "app: nf12stub-ocstub-py"
    configpatch:
      - applyTo: NETWORK_FILTER
        filtername: envoy.filters.network.http_connection_manager
        operation: MERGE
        typeconfig: type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionM
anager
        configkey: server_header_transformation
        configvalue: PASS_THROUGH
```

7. Perform helm upgrade on the ocbsf-servicemesh-config release using the modified `ocbsf_custom_values_servicemesh_config_25.2.200.yaml` file.

   ```
   helm upgrade <helm_release_name_for_servicemesh> -n <namespace>
   <servicemesh_charts> -f <servicemesh-custom.yaml>
   ```

   For example,

   ```
   helm upgrade ocbsf-servicemesh-config ocbsf-servicemesh-
   config-25.2.200.tgz -n ocbsf -f
   ocbsf_custom_values_servicemesh_25.2.200.yaml
   ```

8. Configure DNS for Alternate Route service. For more information, see Post-Installation Steps.

3.1.6.5 Enabling Health Check

This section describes how to enable Health Check for ATS.

To enable Health Check, in the `ocats-bsf/values.yaml` file, set the value of `healthcheck` parameter to `true` and enter a valid value to select either **Webscale** or **OCCNE** environment.

To select OCCNE environment, set the `envtype` to **OCCNE** and update the values of the following parameters:

- `Webscale` - Update the value as false
- `envtype` - T0NDTkU= (i.e envtype=$(echo -n '**OCCNE**' | base64))
- `occnehostip` - OCCNE Host IP address
- `occnehostusername` - OCCNE Host Username
- `occnehostpassword` - OCCNE Host Password

To select OCCNE environment, update the values of the following two parameters:

- `Webscale` - Update the value as true

- `envtype` - T0NDTkU= (i.e envtype=$(echo -n 'OCCNE' | base64))

After the configurations are done, encrypt the parameters and provide the values as shown in the following snippet:

```
atsFeatures:  ## DO NOT UPDATE this section without Engineering team's
permission
  healthcheck: true                   # TO enable/disable ATS Health Check.

sshDetails:
  secretname: "healthchecksecret"
  envtype: "T0NDTkU="
  occnehostip: "MTAuMTcuMjE5LjY1"
  occnehostusername: "dXNlcm5hbWU"
  occnehostpassword: "KioqKg=="
```

To select WEBSCALE environment, update the values of the following two parameters:

- `Webscale` - Update the value as true
- `envtype` - V0VCU0NBTEU= (i.e envtype=$(echo -n '**WEBSCALE**' | base64))

After the configurations are done, encrypt the parameters and provide the values as shown in the following snippet:

```
atsFeatures:  ## DO NOT UPDATE this section without Engineering team's
permission
  healthcheck: true                   # TO enable/disable ATS Health Check.

sshDetails:
  secretname: "healthchecksecret"
  envtype: "V0VCU0NBTEU="
  webscalejumpip: "MTAuNzAuMTE3LjQy"
  webscalejumpusername: "dXNlcm5hbWU="
  webscalejumppassword: "KioqKg=="
  webscaleprojectname: "KioqKg=="
  webscalelabserverFQDN: "KioqKg=="
  webscalelabserverport: "KioqKg=="
  webscalelabserverusername: "KioqKg=="
  webscalelabserverpassword: "KioqKg=="
```

> **Note**
>
> Once the ATS is deployed with HealthCheck feature enabled or disabled, then it cannot be changed. To change the configuration, you are required to re-install.

3.1.6.6 Enabling Persistent Volume

> **Note**
>
> The steps provided in this section are optional and required only if Persistent Volume needs be to enabled.

ATS supports Persistent storage to retain ATS historical build execution data, test cases and one-time environment variable configurations. With this enhancement, the user can decide whether to use persistent volume based on their resource requirements. By default, the persistent volume feature is not enabled.

To enable persistent storage, perform the following steps:

1. Create a PVC using PersistentVolumeClaim.yaml file and associate the same to the ATS pod.
   Sample PersistentVolumeClaim.yaml file:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <Enter the PVC Name>
  annotations:
spec:
  storageClassName: <Provide the Storage Class Name>
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <Provide the size of the PV>
```

   a. Set **PersistentVolumeClaim** to the PVC file name.
   b. Enter the **storageClassName** to the Storage Class Name.
   c. Set **storage** to and size of the persistent volume.
      Sample PVC configuration:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: bsf-pvc-25.2.200
annotations:
spec:
  storageClassName: standard
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

2. Run the following command to create PVC:

   ```
   kubectl apply -f <filename> -n <namespace>
   ```

   For example:

   ```
   kubectl apply -f PersistentVolumeClaim.yaml -n ocbsf
   ```

   Output:

   ```
   persistentvolumeclaim/bsf-pvc-25.2.200 created
   ```

3. Once the PVC is created, run the following command to verify that it is bound to the namespace and is available.

   ```
   kubectl get pvc -n <namespace used for pvc creation>
   ```

   For example:

   ```
   kubectl get pvc -n ocbsf
   ```

   Sample output:

   ```
   NAME              STATUS        VOLUME
   CAPACITY  ACCESS MODES   STORAGECLASS   AGE
   bsf-pvc-25.2.200   Bound    pvc-65484045-3805-4064-9fc3-f9eeeaccc8b8
   1Gi        RWO            standard      11s
   ```

   Verify that the `STATUS` is `Bound` and rest of the parameters like `NAME`, `CAPACITY`, `ACCESS MODES`, and `STORAGECLASS` are as mentioned in the `PersistentVolumeClaim.yaml` file.

   > **Note**
   >
   > Do not proceed further with the next step if there is an issue with the PV creation and contact your administrator to get the PV Created.

4. Enable PVC:
   a. Set the **PVEnabled** flag to **true**.
   b. Set **PVClaimName** to the PVC created in Step 1.

   ```
   PVEnabled: true
   PVClaimName: "ocbsf-pvc-25.2.200"
   ```

   > **Note**
   >
   > Make sure that ATS is deployed before proceeding to the further steps.

5. Copy the <nf_main_folder> and <jenkins jobs> folders from the tar file to their ATS pod and restart the pod.
   a. Extract the tar file.

      ```
      tar -xvf ocats-bsf-data-25.2.200.tgz
      ```

   b. Run the following commands to copy the desired folder.

      ```
      kubectl cp ocats-bsf-data-25.2.200/ocbsf_tests <namespace>/<pod-name>:/var/lib/jenkins/
      ```

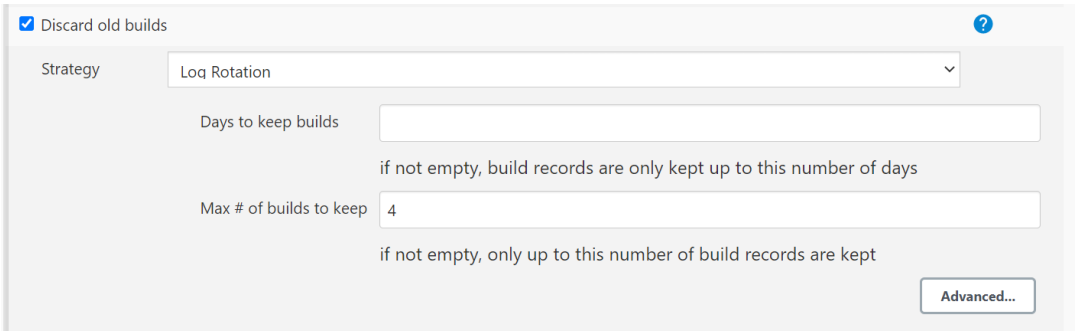
      ```
      kubectl cp ocats-bsf-data-25.2.200/jobs <namespace>/<pod-name>:/var/lib/jenkins/.jenkins/
      ```

   c. Restart the pod.

      ```
      kubectl delete po <pod-name> -n <namespace>
      ```

6. Once the Pod is up and running, log in to the Jenkins console and configure the **Discard old Builds** option to configure the number of Jenkins builds, which must be retained in the persistent volume.

Figure 3-1 Discarding Old Builds



> **Note**
>
> If **Discard old Builds** is not configured, Persistent Volume can get filled when there are huge number of builds.

For more details on Persistent Volume Storage, see Persistent Volume for 5G ATS.

3.1.6.7 ATS-BSF API Extended Support

The ATS application programming interface (API) feature provides APIs, to perform routine ATS tasks such as starting the ATS suite, monitoring and stopping the ATS suite etc.

By default, this feature is enabled in `values.yaml` file.

```
atsFeatures:
  atsApi: true
```

For more details about the ATS API feature, refer to ATS API section.

This ATS feature is extended to provide the ability of running single features, or scenarios, or stages, or groups, or based on tags execution using the API. This also allows running of test cases by providing the features, or scenarios, or stages, or groups, or tags in the `curl` request to the server.

For more details about the API interfaces, refer to Use the RESTful Interfaces section.

3.1.6.8 Configuration for Dual Stack Support

Use the below configuration to support Dual Stack deployments of ocats-bsf:

```
# Possible values : IPv4, IPv6, IPv4_IPv6,IPv6_IPv4,ClusterPreferred
deploymentMode: ClusterPreferred
```

> **Note**
>
> For release 25.2.200, do not change the value of **deploymentMode**. Set it as **ClusterPreferred** only.

To enable IPv6 or Dual Stack configurations for ATS and stubs, see Post-Installation Steps.

3.1.7 Deploying ATS and Pods

3.1.7.1 Deploying ATS in Kubernetes Cluster

> **Important**
>
> This Procedure is for Backwards porting purpose only and should not be considered as the Subsequent Release POD Deployment Procedure.

Prerequisite: Make sure that the old PVC, which contains the old release POD data is available.

To deploy ATS, perform the following steps:

1. Run the following command to deploy ATS using the updated helm charts:

> **Note**
>
> Ensure that all the the components, that is, ATS, stub pods and CNC BSF are deployed in the same namespace.

Using Helm

```
helm install -name <release_name> ocats-bsf-25.2.200.tgz --namespace
<namespace_name> -f <values-yaml-file>
```

For example:

```
helm install -name ocats ocats-bsf-25.2.200.tgz --namespace ocbsf -f ocats-
bsf/values.yaml
```

2. Run the following command to verify ATS deployment:

```
helm ls -n ocbsf
```

The output of the command is as follows:

```
NAME                    REVISION        UPDATED
STATUS          CHART                   APP VERSION     NAMESPACE
ocats                   1               Mon Nov 14 14:56:11 2020
DEPLOYED        ocats-bsf-25.2.200      25.2.200.0.0       ocbsf
```

If the deployment is successful, the status is **Deployed**.

Troubleshooting Issues with Logging in to ATS GUI

In case of any issue with logging in to ATS GUI using valid credentials, perform the following steps:

1. Log in to the ATS pod.
2. Create a shell script with following commands inside ATS pod:

```
USERS_DIR="/var/lib/jenkins/.jenkins/jobs/users"

for f in "$USERS_DIR"/*; do
  base=$(basename "$f")
  new="${base%%_*}"
  [ "$base" != "$new" ] && mv "$f" "$USERS_DIR/$new"
done

ps -ef | grep 8080 | grep jenkins | awk '{ print $2 }' | head -1 | xargs -
I {} kill "{}"

java -Djava.awt.headless=true -
Dhudson.remoting.ClassFilter=org.jenkinsci.plugins.testresultsanalyzer.Test
ResultsAnalyzerExtension\$DescriptorImpl -
Djenkins.install.runSetupWizard=false -
Dorg.jenkinsci.plugins.pipeline.modeldefinition.parser.RuntimeASTTransforme
r.SCRIPT_SPLITTING_TRANSFORMATION=true -
Dorg.jenkinsci.plugins.workflow.cps.LoggingInvoker.fieldSetWarning=false -
Dorg.jenkinsci.plugins.pipeline.modeldefinition.parser.RuntimeASTTransforme
r.SCRIPT_SPLITTING_ALLOW_LOCAL_VARIABLES=true -jar /usr/share/java/
jenkins.war --httpPort=8080 > /var/lib/jenkins/.jenkins/jenkins.log 2>&1 &
```

3. Run the following command to grant execute permission:

   ```
   chmod 777 <script>.sh
   ```

4. Run the following command to execute the shell script:

   ```
   ./<script>.sh
   ```

   After the execution is complete, ATS restarts automatically.

5. Try logging in to the ATS GUI with the given credentials.

3.1.7.2 Deploying ocstub-py Stubs in Kubernetes Cluster

> **Note**
>
> Enabling pre-defined priming configuration is mandatory for nf1stub and nf11stub only. (Please do not enable it in nf12stub stub.)

To deploy ocstub-py stubs with pre-defined priming configurations, perform the following steps:

1. Navigate to `ocats-bsf-tools-25.2.200.0.0` folder and run the following command:

   ```
   tar -zxvf ocstub-pkg-25.2.202.0.0.tgz
   ```

   The output of the command shows:

   - `ocstub-py-25.2.202.tgz`
   - `ocstub-py-image-25.2.202.tar`

2. Deploy the additional stubs required to validate the session retry feature.
   You can use `nf11stub` or `nf12stub` as alternte FQDN for `nf1stub`.

   a. Run the following command to load the stub image.

      ```
      docker load --input ocstub-py-image-25.2.202.tar
      ```

   b. Tag and push the image to your docker registry using below commands.

      ```
      docker tag ocstub-py:25.2.202 localhost:5000/ocstub-py:25.2.202
      docker push localhost:5000/ocstub-py:25.2.202
      ```

   c. Untar the helm charts `ocstub-py-25.2.202.tgz` and update the registry name, image name and tag (if required) in `ocstub-py/values.yaml` file.

      ```
      tar -zxvf ocstub-py-25.2.202.0.0.tgz
      ```

> **Note**
>
> From 24.2.0 onwards, service port names are configurable in ocstub-py. But as per Istio standard, it's advisable to keep the default values as it as.
> Example:
>
> ```
> names:
>   http: "http"
>   h2c: "http2-h2c"
>   h2: "http2-h2"
> ```

d. Configuration for Dual Stack Support:
Use the below configuration to support Dual Stack deployments of ocstub-py:

```
# The below section is to support Dual Stack Deployments of ocstub
# ipFamilies : IPv4, IPv6, IPv4_IPv6, IPv6_IPv4, ClusterPreferred
deploymentMode: &deploymentMode ClusterPreferred
```

> **Note**
>
> For release 25.2.200, do not change the value of **deploymentMode**. Set it as **ClusterPreferred**only.

To enable IPv6 or Dual Stack configurations for ATS and stubs, see Post-Installation Steps.

e. If required, change `apiVersion` to `apps/v1` in `ocstub-py/templates/deployment.yaml` file.

```
apiVersion: apps/v1
```

> **Note**
>
> If the support for pre-defined priming feature is required, perform the following steps to configure pre-defined priming.

f. Copy `ocstub-py/values.yaml` file to a new file with name `pre_priming_values.yaml`.

g. Edit the `ocstub-py/pre_priming_values.yaml` file.

h. Set the value of `preConfig` flag to `true` and replace the default configuration with below configurations under `predefined_prime_configuration` section.
Pre-defined Priming Configuration:

```
preConfig:
  enabled: true
  predefined_prime_configuration: |+
    [
```

```
      {
        "method": "GET",
        "statuscode": "200",
        "url": "/nnrf-nfm/v1/nf-instances/fe7d992b-0541-4c7d-ab84-
c6d70b1b0666",
        "data": "{\"nfInstanceId\": \"fe7d992b-0541-4c7d-ab84-
c6d70b1b0666\", \"nfType\": \"BSF\", \"nfStatus\": \"REGISTERED\",
\"heartBeatTimer\": 2, \"fqdn\": \"ocbsf1-2-api-
gateway.bsf1-2.svc.atlantic.morrisville.us.lab.oracle.com\",
\"priority\": 1, \"capacity\": 1, \"load\": 2, \"bsfInfo\":
{\"ipv4AddressRanges\": [{\"start\": \"10.0.0.1\", \"end\":
\"10.113.255.255\"}], \"ipv6PrefixRanges\": [{\"start\":
\"2800:a00:cc03::/64\", \"end\": \"2800:a00:cc04::/64\"}]},
\"nfServices\": [{\"serviceInstanceId\": \"03063893-
cf9e-4f7a-9827-111111111111\", \"serviceName\": \"nbsf-management\",
\"versions\": [{\"apiVersionInUri\": \"v1\", \"apiFullVersion\":
\"1.0.0\"}], \"scheme\": \"http\", \"nfServiceStatus\": \"REGISTERED\",
\"fqdn\": \"ocbsf1-2-api-
gateway.bsf1-2.svc.atlantic.morrisville.us.lab.oracle.com\",
\"interPlmnFqdn\": null, \"ipEndPoints\": [{\"ipv4Address\":
\"10.233.22.149\", \"transport\": \"TCP\", \"port\": 80}],
\"apiPrefix\": null, \"allowedNfTypes\": [\"PCF\", \"AF\", \"NEF\"],
\"priority\": 1, \"capacity\": 1, \"load\": 2}]}",
        "headers": "{\"Content-Type\": \"application/json\"}"
      },
      {
        "method": "PUT",
        "statuscode": "201",
        "url": "/nnrf-nfm/v1/nf-instances/fe7d992b-0541-4c7d-ab84-
c6d70b1b0666",
        "data": "{\"nfInstanceId\": \"fe7d992b-0541-4c7d-ab84-
c6d70b1b0666\", \"nfType\": \"BSF\", \"nfStatus\": \"REGISTERED\",
\"heartBeatTimer\": 30, \"fqdn\": \"ocbsf1-2-api-
gateway.bsf1-2.svc.atlantic.morrisville.us.lab.oracle.com\",
\"priority\": 1, \"capacity\": 1, \"load\": 2, \"bsfInfo\":
{\"ipv4AddressRanges\": [{\"start\": \"10.0.0.1\", \"end\":
\"10.113.255.255\"}], \"ipv6PrefixRanges\": [{\"start\":
\"2800:a00:cc03::/64\", \"end\": \"2800:a00:cc04::/64\"}]},
\"nfServices\": [{\"serviceInstanceId\": \"03063893-
cf9e-4f7a-9827-111111111111\", \"serviceName\": \"nbsf-management\",
\"versions\": [{\"apiVersionInUri\": \"v1\", \"apiFullVersion\":
\"1.0.0\"}], \"scheme\": \"http\", \"nfServiceStatus\": \"REGISTERED\",
\"fqdn\": \"ocbsf1-2-api-
gateway.bsf1-2.svc.atlantic.morrisville.us.lab.oracle.com\",
\"interPlmnFqdn\": null, \"ipEndPoints\": [{\"ipv4Address\":
\"10.233.22.149\", \"transport\": \"TCP\", \"port\": 80}],
\"apiPrefix\": null, \"allowedNfTypes\": [\"PCF\", \"AF\", \"NEF\"],
\"priority\": 1, \"capacity\": 1, \"load\": 2}]}",
        "headers": "{\"Content-Type\": \"application/json\"}"
      },
      {
        "method": "PATCH",
        "statuscode": "204",
        "url": "/nnrf-nfm/v1/nf-instances/fe7d992b-0541-4c7d-ab84-
c6d70b1b0666",
        "data": "{}",
```

```
        "headers": "{\"Content-Type\": \"application/json\"}"
      },
      {
        "method": "POST",
        "statuscode": "201",
        "url": "/nnrf-nfm/v1/subscriptions",
        "data": "{\"nfStatusNotificationUri\": \"http://ocbsf-ocbsf-
ingress-gateway.ocpcf.svc/nnrf-client/v1/notify\", \"reqNfType\":
\"BSF\", \"subscriptionId\": \"2d77e0de-15a9-11ea-8c5b-b2ca002e6839\",
\"validityTime\": \"2050-12-26T09:34:30.816Z\"}",
        "headers": "{\"Content-Type\": \"application/json\"}"
      }
    ]
```

> **Note**
>
> - The `predefined_prime_configuration` contains variables such as `nfInstanceId`, `nfType`, and `fqdn` in the data's content. Make sure to verify and update the variables based on the payload message that must be included in the response from the NRF on a request.
> - The default value of `nfInstanceId` variable is `fe7d992b-0541-4c7d-ab84-c6d70b1b0666`.

i. Run the following command to deploy the stub:

```
helm install -name <release_name> ocstub-py --set env.NF=<NF> --set
env.LOG_LEVEL=<DEBUG/INFO> --set service.name=<service_name>--set
service.appendReleaseName=false --set preConfig.enabled=false --
namespace=<namespace_name> -f <values.yaml-file>
```

Install nf1stub and nf11stub with updated *ocstub-py/pre_priming_values.yaml* file.

```
helm install -name nf1stub ocstub-py --set env.NF=BSF --set
env.LOG_LEVEL=DEBUG --set service.name=nf1stub --set
service.appendReleaseName=false --set preConfig.enabled=true --
namespace=ocbsf -f ocstub-py/values.yaml
```

```
helm install -name nf11stub ocstub-py --set env.NF=BSF --set
env.LOG_LEVEL=DEBUG --set service.name=nf11stub --set
service.appendReleaseName=false --set preConfig.enabled=true --
namespace=ocbsf -f ocstub-py/values.yaml
```

Install nf12stub with the default *ocstub-py/pre_priming_values.yaml* file

```
helm install -name nf12stub ocstub-py --set env.NF=BSF --set
env.LOG_LEVEL=DEBUG --set service.name=nf12stub --set
service.appendReleaseName=false --set preConfig.enabled=false --
namespace=ocbsf -f ocstub-py/values.yaml
```

j. Run the following command to verify the stub deployment:

```
helm ls -n ocbsf
```

Sample output:

```
NAME                    REVISION            UPDATED
STATUS          CHART                   APP VERSION
NAMESPACE
nf11stub            1               Thu Jul  29 05:55:48 2024
DEPLOYED        ocstub-py-25.2.202      25.2.202.0.0            ocbsf
nf12stub            1               Thu Jul  29 05:55:50 2024
DEPLOYED        ocstub-py-25.2.202      25.2.202.0.0            ocbsf
nf1stub             1               Thu Jul  29 05:55:47 2024
DEPLOYED        ocstub-py-25.2.202      25.2.202.0.0            ocbsf
```

k. Run the following command to verify the ATS and Stubs deployment status:

```
helm status -n ocbsf
```

l. Run the following command to verify if all the services are installed.

```
kubectl get po -n ocbsf
```

Sample output:

```
NAME                                                  READY
STATUS    RESTARTS   AGE
nf11stub-ocstub-py-7bffd6dcd7-ftm5f                   1/1     Running
0          3d23h
nf12stub-ocstub-py-547f7cb99f-7mpll                   1/1     Running
0          3d23h
nf1stub-ocstub-py-bdd97cb9-xjrkx                      1/1     Running
0          3d23h
```

3. Verify the changes related to stub pre-defined prime configuration.

a. Run the following command to verify the status of all the config-map.

```
kubectl get cm -n ocbsf
```

Notice the change in the number of config-map counts. There will be one extra config-maps of nf1stub and nf11stub.

A sample of config-map with pre-defined priming is as follows:

```
NAME                 DATA   AGE
cm-pystub-nf1stub               1      3h35m
cm-pystub-nf11stub               1      3h35m
```

Updating the Predefined priming configurations

> **Note**
>
> This procedure is applicable only when Predefined priming configuration is enabled.

Post the deployment, there might be the chance to modify some values inside the stub pre-priming configurations, for example, nfInstanceID in the url.

1. Run the following command to edit the config-map of nf1stub:

   ```
   kubectl get cm -n ocbsf


   kubectl edit cm cm-pystub-nf1stub -n ocbsf
   ```

2. Edit the configurations as required and save it.
3. Restart the nf1stub pod.
4. Verify the log to confirm the changes.
5. Follow the same procedure for nf11stub pod.

3.1.7.3 Deploying DNS Stub in Kubernetes Cluster

> **Note**
>
> Ensure there are sufficient resources and limit for DNS Stub. Set the resource request and limit values in the **resources** section of the **values.yaml** file as follows:
>
> ```
> resources: {}
>   # We usually recommend not to specify default resources and to leave
> this as a conscious
>   # choice for the user. This also increases chances charts run on
> environments with little
>   # resources, such as Minikube. If you do want to specify resources,
> uncomment the following
>   # lines, adjust them as necessary, and remove the curly braces after
> 'resources:'.  # limits:
>   #  cpu: 1000m
>   #  memory: 1024Mi
>   # requests:
>   #  cpu: 500m
>   #  memory: 500Mi
> ```

To deploy DNS Stub in Kubernetes cluster, perform the following steps:

1. Go to the `ocats-bsf-tools-25.2.200.0.0` folder and run the following command to extract the ocstub tar file content:
   `tar -zxvf ocdns-pkg-25.2.204.0.0.tgz`

Sample output:

```
[cloud-user@platform-bastion-1 ocdns-pkg-25.2.204.0.0]$ ls -ltrh
total 211M
-rw-------. 1 cloud-user cloud-user 211M Mar 14 14:49 ocdns-bind-
image-25.2.204.tar
-rw-r--r--. 1 cloud-user cloud-user 2.9K Mar 14 14:49 ocdns-
bind-25.2.204.tgz
```

2. Run the following command in your cluster to load the DNS STUB image:
   `docker load --input ocdns-bind-image-25.2.204.tar`
3. Run the following commands to tag and push the DNS STUB image:

   ```
   docker tag ocdns-bind:25.2.204 localhost:5000/ocdns-bind:25.2.204
   docker push localhost:5000/ocdns-bind:25.2.204
   ```

4. Run the following command to untar the helm charts, `ocdns-bind-25.2.204.tgz`.
   `tar -zxvf ocdns-bind-25.2.204.tgz`
5. Update the registry name, image name and tag (if required) in the `ocdns-bind/values.yaml` file as required. For this, open the **values.yaml** file and update the `image.repository` and `image.tag` parameters.
6. Run the following command to deploy the DNS Stub.
   Using Helm:

   ```
   helm install -name ocdns ocdns-bind-25.2.204.tgz --namespace ocbsf -f
   ocdns-bind/values.yaml
   ```

7. Capture the `cluster name` of the deployment, `namespace` where nfstubs are deployed, and the cluster IP of DNS Stub.

   To capture the DNS Stub cluster IP:

   ```
   kubectl get svc -n ocbsf | grep dns
   ```

   Sample output:

   ```
   [cloud-user@platform-bastion-1 ocdns-pkg-25.2.204.0.0]$ kubectl get svc -n
   ocbsf | grep dns
   NAME      TYPE           CLUSTER-IP      EXTERNAL-IP
   PORT(S)               AGE
   ocdns     ClusterIP      10.233.11.45    <none>          53/
   UDP,6236/TCP        19h
   ```

   To capture the cluster name:

   ```
   kubectl -n kube-system get configmap kubeadm-config -o yaml | grep
   clusterName
   ```

Sample output:

```
clusterName: platform
```

3.1.7.4 Deploying ocdiam Simulator in Kubernetes Cluster

To deploy ocdiam Simulator in Kubernetes cluster, perform the following steps:

1. Go to the `ocats-bsf-tools-25.2.200.0.0` folder and run the following command to extract the ocstub tar file content:
   `tar -zxvf ocdiam-pkg-25.2.204.0.0.tgz`

   Sample output:

   ```
   [cloud-user@platform-bastion-1 ocdiam-pkg-25.2.204.0.0]$ ls -ltrh
   total 908M
   -rw-------. 1 cloud-user cloud-user 908M Mar 14 14:49 ocdiam-sim-
   image-25.2.204.tar
   -rw-r--r--. 1 cloud-user cloud-user 3.8K Mar 14 14:49 ocdiam-
   sim-25.2.204.tgz
   ```

2. Run the following command in your cluster to load the Diameter Simulator image:

   ```
   docker load --input ocdiam-sim-image-25.2.204.tar
   ```

3. Run the following commands to tag and push the Diameter Simulator image:

   ```
   docker tag ocdiam-sim:25.2.204 localhost:5000/ocdiam-sim:25.2.204
   docker push localhost:5000/ocdiam-sim:25.2.204
   ```

4. Run the following command to untar the helm charts, `ocdiam-sim-25.2.204.tgz`.

   ```
   tar -zxvf ocdiam-sim-25.2.204.tgz
   ```

5. Update the registry name, image name and tag (if required) in the `ocdiam-sim/values.yaml` file as required. For this, open the **values.yaml** file and update the `image.repository` and `image.tag` parameters.
6. Run the following command to deploy the Diameter Simulator.
   Using Helm:

   ```
   helm install -name ocdiam-sim ocdiam-sim --namespace ocbcf -f ocdiam-sim/
   values.yaml
   ```

   Output:

   ```
   ocdiam-sim-69968444b6-fg6ks               1/1     Running   0
   5h47m
   ```

Sample of BSF namespace with BSF and ATS after installation:

```
[cloud-user@platform-bastion-1 ocstub-pkg-25.2.202.0.0]$ kubectl get po -n
ocbsf
NAME                                                    READY   STATUS
RESTARTS   AGE
```

```
ocbsf-appinfo-6fc99ffb85-f96j2                          1/1     Running
1          3d23h
ocbsf-bsf-management-service-df6b68d75-m77dv            1/1     Running
0          3d23h
ocbsf-oc-config-79b5444f49-7pwzx                        1/1     Running
0          3d23h
ocbsf-oc-diam-connector-77f7b855f4-z2p88                1/1     Running
0          3d23h
ocbsf-oc-diam-gateway-0                                 1/1     Running
0          3d23h
ocbsf-ocats-bsf-5d8689bc77-cxdvx                        1/1     Running
0          3d23h
ocbsf-ocbsf-egress-gateway-644555b965-pkxsb             1/1     Running
0          3d23h
ocbsf-ocbsf-ingress-gateway-7558b7d5d4-lfs5s            1/1     Running
4          3d23h
ocbsf-ocbsf-nrf-client-nfmanagement-d6b955b48-4pptk     1/1     Running
0          3d23h
ocbsf-ocdns-ocdns-bind-75c964648-j5fsd                  1/1     Running
0          3d23h
ocbsf-ocpm-cm-service-7775c76c45-xgztj                  1/1     Running
0          3d23h
ocbsf-ocpm-queryservice-646cb48c8c-d72x4                1/1     Running
0          3d23h
ocbsf-performance-69fc459ff6-frrvs                      1/1     Running
4          3d23h
ocbsfnf11stub-7bffd6dcd7-ftm5f                          1/1     Running
0          3d23h
ocbsfnf12stub-547f7cb99f-7mpll                          1/1     Running
0          3d23h
ocbsfnf1stub-bdd97cb9-xjrkx                             1/1     Running
0          3d23h
ocdiam-sim-69968444b6                                   1/1     Running
0          3d23h
```

3.1.8 Post-Installation Steps

The section describes post-installation steps that users should perform after deploying ATS and stub pods.

Enabling IPv6 for ATS

Perform this procedure after ATS for BSF is installed. This procedure creates services with the required IP family configuration.

Below are the prerequisites for this procedure:

- Ensure that the Kubernetes cluster supports IPv6 or dual stack networking.
- Do not edit an existing service to change its IP family from IPv4 to IPv6.
- Create a temporary directory to store the service YAML file exported in Step 1.

1. Run the following command to export the existing service YAML file:

   ```
   kubectl get svc <SERVICE_NAME> -n <NAMESPACE> -o yaml > /tmp/<directory>/
   <my-svc.yaml>
   ```

The current service definition is retrieved and written to `/tmp/<directory>/<my-svc.yaml>`.

Sample output:

```
kubectl get svc nf11stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/
nf11stub.yaml
kubectl get svc nf12stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/
nf12stub.yaml
kubectl get svc nf1stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/
nf1stub.yaml
kubectl get svc ocats-bsf -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocats-
bsf.yaml
kubectl get svc ocdiam-sim-1 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
sim-1.yaml
kubectl get svc ocdiam-sim-2 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
sim-2.yaml
kubectl get svc ocdiam-sim-3 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
sim-3.yaml
kubectl get svc ocdiam-sim-4 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
sim-4.yaml
kubectl get svc ocdns-stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdns-
stub.yaml
```

2. Run the following command to remove the existing service:

   ```
   kubectl delete svc <SERVICE_NAME> -n <NAMESPACE>
   ```

   Sample output:

   ```
   kubectl delete svc nf11stub -n ocbsf
   kubectl delete svc nf12stub -n ocbsf
   kubectl delete svc nf1stub -n
   ocbsf
   kubectl delete svc ocats-bsf -n ocbsf
   kubectl delete svc ocdiam-sim-1 -n ocbsf
   kubectl delete svc ocdiam-sim-2 -n ocbsf
   kubectl delete svc ocdiam-sim-3 -n ocbsf
   kubectl delete svc ocdiam-sim-4 -n ocbsf
   kubectl delete svc ocdns-stub -n ocbsf
   ```

3. Edit the exported YAML file and remove the value of the following fields if they exist:

   ```
   clusterIP:
   clusterIPs:
   ```

   Unedited YAML file:

   ```
   spec:
     clusterIP: 10.233.13.190
     clusterIPs:
     - 10.233.13.190
     internalTrafficPolicy: Cluster
   ```

```
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
```

Edited YAML file:

```
spec:
  clusterIP:
  clusterIPs:
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv6
  ipFamilyPolicy: SingleStack
```

> **Note**
>
> Do not specify the value of `clusterIP` or `clusterIPs`. These values are automatically assigned by Kubernetes.

4. Run the following command to recreate the service with the updated YAML file:

   ```
   kubectl apply -f my-svc.yaml
   ```

   The service is created with the required IP family configuration.

   Sample output:

   ```
   [cnpolicy@bastion-1 svcs]$ pwd
   /tmp/ocbsf/svcs

   [cnpolicy@bastion-1 svcs]$ ls
    nf12stub.yaml         ocdiam-sim-1.yaml  ocdiam-sim-3.yaml  ocdns-
   stub.yaml
   nf11stub.yaml                      nf1stub.yaml      ocats-bsf.yaml
   ocdiam-sim-2.yaml  ocdiam-sim-4.yaml

   kubectl apply -f nf11stub.yaml -n ocbsf
   kubectl apply -f nf12stub.yaml -n ocbsf
   kubectl apply -f nf1stub.yaml -n
   ocbsf
   kubectl apply -f ocats-bsf.yaml -n ocbsf
   kubectl apply -f ocdiam-sim-1.yaml -n ocbsf
   kubectl apply -f ocdiam-sim-2.yaml -n ocbsf
   kubectl apply -f ocdiam-sim-3.yaml -n ocbsf
   kubectl apply -f ocdiam-sim-4.yaml -n ocbsf
   kubectl apply -f ocdns-stub.yaml -n ocbsf
   ```

5. To verify if the service has an IPv6 address, run the following command:

   ```
   kubectl get svc -n <NAMESPACE>
   ```

Sample output:

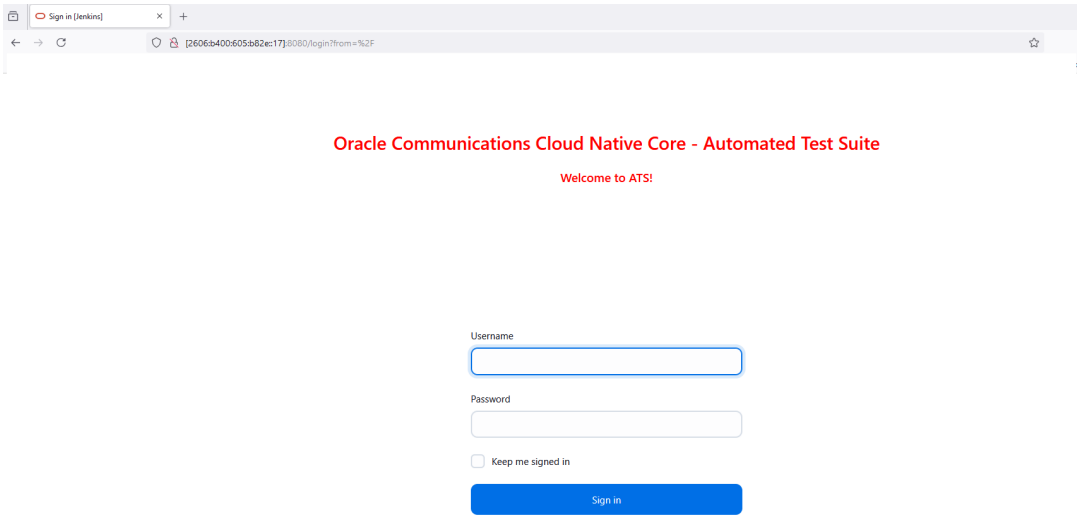
```
NAME                                    TYPE         CLUSTER-IP
EXTERNAL-IP
PORT(S)                                        AGE
-------------------------------------------------------------------------------
-----------------------------------------------
nf11stub                                  ClusterIP
fd00:0:0:2::74aa     <none>                              8080/
TCP,8081/TCP                           8h
nf12stub                                  ClusterIP
fd00:0:0:2::10cf     <none>                              8080/
TCP,8081/TCP                           8h
nf1stub                                   ClusterIP
fd00:0:0:2::cea4     <none>                              8080/
TCP,8081/TCP                           8h
ocbsf-altsvc-cache                        ClusterIP
None                 <none>
8000/TCP                                     9h
ocbsf-egw-cache                           ClusterIP
None                 <none>
8000/TCP                                     9h
ocbsf-igw-cache                           ClusterIP
None                 <none>
8000/TCP                                     9h
ocbsf-ocbsf-alternate-route               ClusterIP
fd00:0:0:2::4735     <none>
8000/TCP                                     9h
ocbsf-ocbsf-app-info                      ClusterIP
fd00:0:0:2::7a03     <none>
8000/TCP                                     9h
ocbsf-ocbsf-audit                         ClusterIP
fd00:0:0:2::3be2     <none>
8000/TCP                                     9h
ocbsf-ocbsf-bsf-management                ClusterIP
fd00:0:0:2::9f7f     <none>                              5903/
TCP,8443/TCP                           9h
ocbsf-ocbsf-config-mgmt                   LoadBalancer
fd00:0:0:2::bc02     2606:b400:605:b82e::7
8000:31125/TCP                               9h
ocbsf-ocbsf-config-server                 ClusterIP
fd00:0:0:2::8746     <none>
8000/TCP                                     9h
ocbsf-ocbsf-egress-gateway                ClusterIP
fd00:0:0:2::c15f     <none>
8000/TCP                                     9h
ocbsf-ocbsf-ingress-gateway               LoadBalancer
fd00:0:0:2::a5a6     2606:b400:605:b82e::8
80:32507/TCP                                 9h
ocbsf-ocbsf-ingress-gateway-intra-nf      ClusterIP
fd00:0:0:2::1237     <none>
8008/TCP                                     9h
ocbsf-ocbsf-nrf-client-cache              ClusterIP
None                 <none>
5907/TCP                                     9h
ocbsf-ocbsf-nrf-client-nfdiscovery        ClusterIP
```

```
fd00:0:0:2::d883     <none>                                8000/
TCP,9443/TCP                                  9h
ocbsf-ocbsf-nrf-client-nfmanagement          ClusterIP
fd00:0:0:2::2963     <none>                                8000/
TCP,9443/TCP                                  9h
ocbsf-ocbsf-oc-diam-gateway                  LoadBalancer
fd00:0:0:2::bf45     2606:b400:605:b82e::12
3868:31695/TCP                                   9h
ocbsf-ocbsf-oc-diam-gateway-coherence-headless ClusterIP
None                 <none>
5801/TCP                                         9h
ocbsf-ocbsf-oc-diam-gateway-headless         ClusterIP
None                 <none>                                3868/
TCP,8000/TCP                                  9h
ocbsf-ocbsf-oc-diam-gateway-internal         ClusterIP
fd00:0:0:2::26df     <none>
8000/TCP                                         9h
ocbsf-ocbsf-perf-info                        ClusterIP
fd00:0:0:2::8e23     <none>
8000/TCP                                         9h
ocbsf-ocbsf-query                            ClusterIP
fd00:0:0:2::5075     <none>
8000/TCP                                         9h
ocats-bsf                                    LoadBalancer
fd00:0:0:2::62dc     2606:b400:605:b82e::17            8080:30942/
TCP,5001:31901/TCP,3868:32662/TCP    8h
ocdiam-sim-1                                 ClusterIP
fd00:0:0:2::a25b     <none>                                8089/
TCP,3868/TCP                                  8h
ocdiam-sim-2                                 ClusterIP
fd00:0:0:2::d3db     <none>                                8089/
TCP,3868/TCP                                  8h
ocdiam-sim-3                                 ClusterIP
fd00:0:0:2::b656     <none>                                8089/
TCP,3868/TCP                                  8h
ocdiam-sim-4                                 ClusterIP
fd00:0:0:2::d30f     <none>                                8089/
TCP,3868/TCP                                  8h
ocdns-stub                                   ClusterIP
fd00:0:0:2::2341     <none>                                53/
UDP,6236/TCP                                  8h
```

> **Note**
>
> Ensure that only IPv6 addresses are observed.

6. To access the ATS GUI with an IPv6, the IP in IPv6 of ocats and the port 8080 are entered in the browser URL of ATS (Jenkins).

Figure 3-2 Accessing ATS GUI using IPv6



Enabling Dual Stack for ATS

Perform the following procedure to enable dual stack support for ATS.

Modify the IP address type to IPv6 dual stack or IPv4 dual stack by following these steps:

Below are the prerequisites for this procedure:

- Ensure that the Kubernetes cluster supports IPv6 or dual stack networking.
- Do not edit an existing service to change its IP family from IPv4 to IPv6.
- Create a temporary directory to store the service YAML file exported in Step 1.

1. Run the following command to export the existing service YAML file:

   ```
   kubectl get svc <SERVICE_NAME> -n <NAMESPACE> -o yaml > /tmp/<directory>/
   <my-svc.yaml>
   ```

   The current service definition is retrieved and written to `/tmp/<directory>/<my-svc.yaml>`.

   Sample output:

   ```
   kubectl get svc nf11stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/
   nf11stub.yaml
   kubectl get svc nf12stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/
   nf12stub.yaml
   kubectl get svc nf1stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/
   nf1stub.yaml
   kubectl get svc ocats-bsf -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocats-
   bsf.yaml
   kubectl get svc ocdiam-sim-1 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
   sim-1.yaml
   kubectl get svc ocdiam-sim-2 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
   sim-2.yaml
   kubectl get svc ocdiam-sim-3 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
   ```

```
sim-3.yaml
kubectl get svc ocdiam-sim-4 -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdiam-
sim-4.yaml
kubectl get svc ocdns-stub -n ocbsf -o yaml > /tmp/ocbsf/svcs/ocdns-
stub.yaml
```

2. Run the following command to remove the existing service:

   ```
   kubectl delete svc <SERVICE_NAME> -n <NAMESPACE>
   ```

   Sample output:

   ```
   kubectl delete svc nf11stub -n ocbsf
   kubectl delete svc nf12stub -n ocbsf
   kubectl delete svc nf1stub -n
   ocbsf
   kubectl delete svc ocats-bsf -n ocbsf
   kubectl delete svc ocdiam-sim-1 -n ocbsf
   kubectl delete svc ocdiam-sim-2 -n ocbsf
   kubectl delete svc ocdiam-sim-3 -n ocbsf
   kubectl delete svc ocdiam-sim-4 -n ocbsf
   kubectl delete svc ocdns-stub -n ocbsf
   ```

3. Edit the exported YAML file and remove the value of the following fields if they exist:

   ```
   clusterIP:
   clusterIPs:
   ```

   Unedited YAML file:

   ```
   spec:
     clusterIP: 10.233.13.190
     clusterIPs:
     - 10.233.13.190
     internalTrafficPolicy: Cluster
     ipFamilies:
     - IPv4
     ipFamilyPolicy: SingleStack
   ```

   Add or modify the `spec` section for IPv6 or IPv4 dual stack:

   ```
   spec:
     ipFamilies:
       - IPv6
       - IPv4
     ipFamilyPolicy: RequireDualStack


   or

   spec:
     ipFamilies:
       - IPv4
   ```

```
    - IPv6
  ipFamilyPolicy: RequireDualStack
```

Edited YAML file:

```
spec:
  clusterIP:
  clusterIPs:
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv6
    - IPv4
  ipFamilyPolicy: RequireDualStack

 or

spec:
  clusterIP:
  clusterIPs:
  internalTrafficPolicy: Cluster
  ipFamilies:
    - IPv4
    - IPv6
  ipFamilyPolicy: RequireDualStack
```

> **Note**
>
> Do not specify the value of `clusterIP` or `clusterIPs`. These values are automatically assigned by Kubernetes.

4. Run the following command to recreate the service with the updated YAML file:

```
kubectl apply -f my-svc.yaml
```

The service is created with the required IP family configuration.

Sample output:

```
[cnpolicy@bastion-1 svcs]$ pwd
/tmp/ocbsf/svcs

[cnpolicy@bastion-1 svcs]$ ls
 nf12stub.yaml         ocdiam-sim-1.yaml  ocdiam-sim-3.yaml  ocdns-
stub.yaml
nf11stub.yaml                       nf1stub.yaml      ocats-bsf.yaml
ocdiam-sim-2.yaml  ocdiam-sim-4.yaml

kubectl apply -f nf11stub.yaml -n ocbsf
kubectl apply -f nf12stub.yaml -n ocbsf
kubectl apply -f nf1stub.yaml -n
ocbsf
kubectl apply -f ocats-bsf.yaml -n ocbsf
kubectl apply -f ocdiam-sim-1.yaml -n ocbsf
kubectl apply -f ocdiam-sim-2.yaml -n ocbsf
```

```
kubectl apply -f ocdiam-sim-3.yaml -n ocbsf
kubectl apply -f ocdiam-sim-4.yaml -n ocbsf
kubectl apply -f ocdns-stub.yaml -n ocbsf
```

5. To verify if the service has a dual stack address with the preference IPv4 or IPv6, run the following command:

```
kubectl get svc -n <NAMESPACE>
```

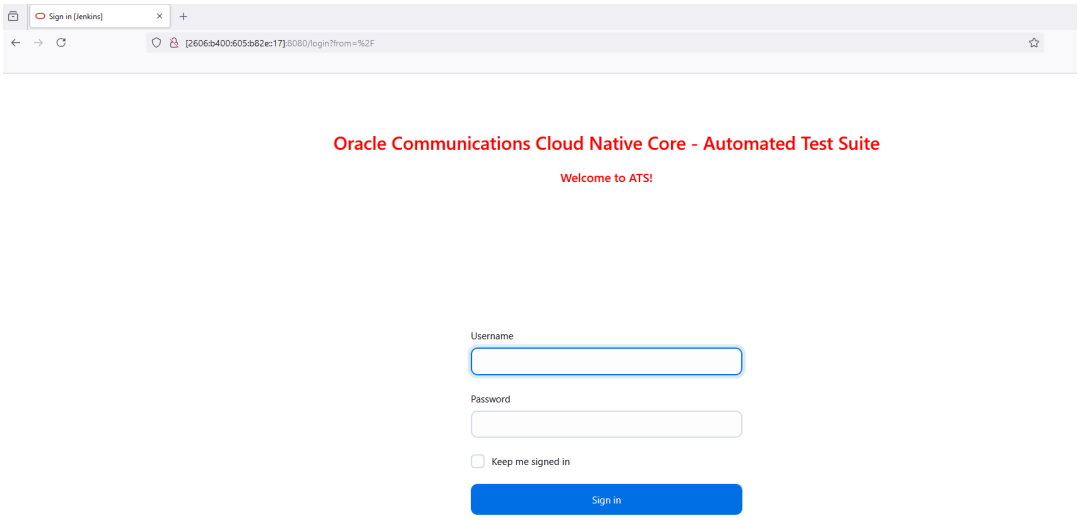
Sample output:

```
NAME                                                 TYPE           CLUSTER-
IP           EXTERNAL-IP
PORT(S)                                            AGE
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------
nf11stub                                              ClusterIP
fd00:0:0:2::74aa     <none>                                          8080/
TCP,8081/TCP                                 8h
nf12stub                                              ClusterIP
fd00:0:0:2::10cf     <none>                                          8080/
TCP,8081/TCP                                 8h
nf1stub                                               ClusterIP
fd00:0:0:2::cea4     <none>                                          8080/
TCP,8081/TCP                                 8h
ocbsf-altsvc-cache                                    ClusterIP
None                 <none>
8000/TCP                                           9h
ocbsf-egw-cache                                       ClusterIP
None                 <none>
8000/TCP                                           9h
ocbsf-igw-cache                                       ClusterIP
None                 <none>
8000/TCP                                           9h
ocbsf-ocbsf-alternate-route                           ClusterIP
fd00:0:0:2::4735     <none>
8000/TCP                                           9h
ocbsf-ocbsf-app-info                                  ClusterIP
fd00:0:0:2::7a03     <none>
8000/TCP                                           9h
ocbsf-ocbsf-audit                                     ClusterIP
fd00:0:0:2::3be2     <none>
8000/TCP                                           9h
ocbsf-ocbsf-bsf-management                            ClusterIP
fd00:0:0:2::9f7f     <none>                                          5903/
TCP,8443/TCP                                 9h
ocbsf-ocbsf-config-mgmt                               LoadBalancer
fd00:0:0:2::bc02     2606:b400:605:b82e::7
8000:31125/TCP                                     9h
ocbsf-ocbsf-config-server                             ClusterIP
fd00:0:0:2::8746     <none>
8000/TCP                                           9h
ocbsf-ocbsf-egress-gateway                            ClusterIP
```

```
fd00:0:0:2::c15f     <none>
8000/TCP                                        9h
ocbsf-ocbsf-ingress-gateway                      LoadBalancer
fd00:0:0:2::a5a6     10.75.206.211,2606:b400:605:b82e::8
80:32507/TCP                                    9h
ocbsf-ocbsf-ingress-gateway-intra-nf             ClusterIP
fd00:0:0:2::1237     <none>
8008/TCP                                        9h
ocbsf-ocbsf-nrf-client-cache                     ClusterIP
None                 <none>
5907/TCP                                        9h
ocbsf-ocbsf-nrf-client-nfdiscovery               ClusterIP
fd00:0:0:2::d883     <none>                                          8000/
TCP,9443/TCP                                 9h
ocbsf-ocbsf-nrf-client-nfmanagement              ClusterIP
fd00:0:0:2::2963     <none>                                          8000/
TCP,9443/TCP                                 9h
ocbsf-ocbsf-oc-diam-gateway                      LoadBalancer
fd00:0:0:2::bf45     10.75.206.210,2606:b400:605:b82e::12
3868:31695/TCP                                  9h
ocbsf-ocbsf-oc-diam-gateway-coherence-headless   ClusterIP
None                 <none>
5801/TCP                                        9h
ocbsf-ocbsf-oc-diam-gateway-headless             ClusterIP
None                 <none>                                          3868/
TCP,8000/TCP                                 9h
ocbsf-ocbsf-oc-diam-gateway-internal             ClusterIP
fd00:0:0:2::26df     <none>
8000/TCP                                        9h
ocbsf-ocbsf-perf-info                            ClusterIP
fd00:0:0:2::8e23     <none>
8000/TCP                                        9h
ocbsf-ocbsf-query                                ClusterIP
fd00:0:0:2::5075     <none>
8000/TCP                                        9h
ocats-bsf                                        LoadBalancer
fd00:0:0:2::62dc     10.75.206.199,2606:b400:605:b82e::13
8080:30942/TCP,5001:31901/TCP,3868:32662/TCP    8h
ocdiam-sim-1                                     ClusterIP
fd00:0:0:2::a25b     <none>                                          8089/
TCP,3868/TCP                                 8h
ocdiam-sim-2                                     ClusterIP
fd00:0:0:2::d3db     <none>                                          8089/
TCP,3868/TCP                                 8h
ocdiam-sim-3                                     ClusterIP
fd00:0:0:2::b656     <none>                                          8089/
TCP,3868/TCP                                 8h
ocdiam-sim-4                                     ClusterIP
fd00:0:0:2::d30f     <none>                                          8089/
TCP,3868/TCP                                 8h
ocdns-stub                                       ClusterIP
fd00:0:0:2::2341     <none>                                          53/
UDP,6236/TCP                                   8h
```

6. To access the ATS GUI with an IPv6, the IP in IPv6 of ocats and the port 8080 are entered in the browser URL of ATS (Jenkins).

Figure 3-3 Accessing ATS GUI using IPv6 in Dual Stack



Alternate Route Service Configurations

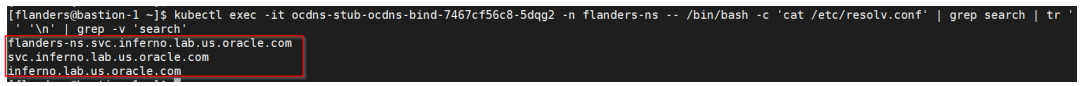
To edit the Alternate Route Service deployment file (ocbcf-ocbsf-alternate-route) that points to DNS Stub, perform the following steps:

1. Run the following command to get searches information from dns-bind pod to enable communication between Alternate Route and dns-bind service:

   ```
   kubectl exec -it <dns-bind pod> -n <NAMESPACE> -- /bin/bash -c 'cat /etc/
   resolv.conf' | grep search | tr ' ' '\n' | grep -v 'search'
   ```

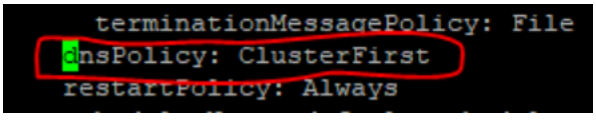
   The following output is displayed after running the command:

   Figure 3-4 Sample Output

   By default alternate service will point to CoreDNS and you will see following settings in deployment file:

   Figure 3-5 Alternate Route Service Deployment File

2. Run the following command to edit the deployment file and add the following content in alternate service to query DNS stub:

   ```
   $kubectl edit deployment ocpcf-occnp-alternate-route -n ocpcf
   ```

a. Add the IP Address of the nameserver that you have recorded after installing the DNS stub (cluster IP Address of DNS Stub).

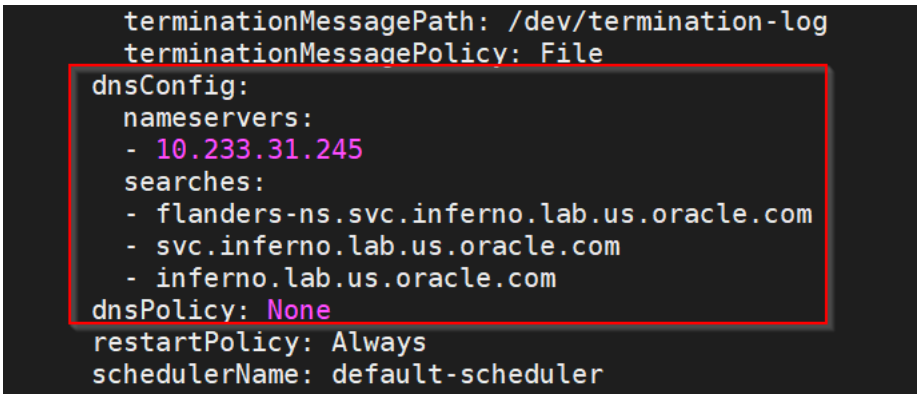
b. Add the search information one by one which you recorded earlier.

c. Set dnsPolicy to "None".

```
dnsConfig:
  nameservers:
  - 10.233.33.169      // cluster IP of DNS Stub
  searches:
  - ocpcf.svc.occne15-ocpcf-ats
  - svc.occne15-ocpcf-ats
  - occne15-ocpcf-ats
dnsPolicy: None
```

For example:

Figure 3-6 Example



NRF client configmap

1. In the **application-config** configmap, configure the following parameters with the respective values:
   - `primaryNrfApiRoot=nf1stub.<namespace_gostubs_are_deployed_in>.svc:8080`
     Example: `primaryNrfApiRoot=nf1stub.ocats.svc:8080`
   - `secondaryNrfApiRoot=nf11stub.<namespace_gostubs_are_deployed_in>.svc:8080`
     Example:`secondaryNrfApiRoot=nf11stub.ocats.svc:8080`
   - `virtualNrfFqdn = nf1stub.<namespace_gostubs_are_deployed_in>.svc`
     Example:`virtualNrfFqdn=nf1stub.ocats.svc`

   > **Note**
   >
   > To get all configmaps in your namespace, run the following command:
   > `kubectl get configmaps -n <BSF_namespace>`

2. (Optional) If persistent volume is used, follow the post-installation steps provided in the Persistent Volume for 5G ATS section.

3.2 Installing ATS for NSSF

This section describes Automated Testing Suite (ATS) installation procedures for Network Slice Selection Function (NSSF) in a cloud native environment. You must perform ATS installation procedures for NSSF in the same sequence as outlined in the following sections.

3.2.1 Resource Requirements

Total Number of Resources

The total number of resource requirements are as follows:

Table 3-5 Total Number of Resources

Resource	CPUs	Memory(GB)	Storage(GB)
NSSF SUT Total	30.2	22	4
cnDBTier Total	40	40	20
ATS Total	14	14	0
Grand Total NSSF ATS	79.2	72	24

Resource Details

The details of resources required to install NSSF-ATS are as follows:

Table 3-6 Resource Details

Microservice	CPUs Required per Pod	Memory Required per Pod (GB)	Storage PVC Required per Pod (GB)	Replicas (regular deployment)	Replicas (ATS deployment)	CPUs Required - Total	Memory Required - Total (GB)	Storage PVC Required - Total (GB)
NSSF Pods								
ingressgateway	4	4	0	2	1	4	4	0
egressgateway	4	4	0	2	1	4	4	0
nsselection	4	2	0	2	1	4	2	0
nsavailability	4	2	0	2	1	4	2	0
nsconfig	2	2	0	1	1	2	2	0
nssubscription	2	2	0	1	1	2	2	0
nrf-client-discovery	1	1	0	2	1	1	1	0
nrf-client-management	1	1	0	1	1	1	1	0
appinfo	0.2	1	0	2	1	0.2	1	0
perfinfo	0.2	0.5	0	1	1	0.2	0.5	0
config-server	0.2	0.5	0	1	1	0.2	0.5	0

Table 3-6 (Cont.) Resource Details

Microservice	CPUs Required per Pod	Memory Required per Pod (GB)	Storage PVC Required per Pod (GB)	Replicas (regular deployment)	Replicas (ATS deployment)	CPUs Required - Total	Memory Required - Total (GB)	Storage PVC Required - Total (GB)
NSSF SUT Totals						**22.6 CPU**	**20 GB**	0
ATS								
ATS Behave	4	4	0	0	1	4	4	0
ATS AMF Stub (Python)	3	3	0	0	1	3	3	0
ATS NRF Stub (Python)	2	2	0	0	1	2	2	0
ATS NRF Stub1 (Python)	2	2	0	0	1	2	2	0
ATS NRF Stub2 (Python)	2	2	0	0	1	2	2	0
OCDNS-BIND	1	1	0	0	1	1	1	0
ATS Totals						**14**	**14**	0
cnDBTier Pods (minimum of 4 worker nodes required)								
vrt-launcher-dt-1.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-dt-2.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-dt-3.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-dt-4.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-mt-1.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-mt-2.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-mt-3.cluster.local	4	4	2	2	1	4	4	2

Table 3-6 (Cont.) Resource Details

Microservice	CPUs Required per Pod	Memory Required per Pod (GB)	Storage PVC Required per Pod (GB)	Replicas (regular deployment)	Replicas (ATS deployment)	CPUs Required - Total	Memory Required - Total (GB)	Storage PVC Required - Total (GB)
vrt-launcher-sq-1.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-sq-2.cluster.local	4	4	2	2	1	4	4	2
vrt-launcher-db-installer.cluster.local	4	4	2	2	1	4	4	2
cnDBTier Totals						**40**	**40**	**20**

3.2.2 Locating and Downloading ATS and Simulator Images

To locate and download the ATS Image from MOS:

1. Log in to My Oracle Support using the appropriate credentials.
2. Select the **Patches and Updates** tab.
3. In the **Patch Search** window, click **Product or Family (Advanced)**.
4. Enter *Oracle Communications Cloud Native Core - 5G* in the **Product** field.
5. Select *Oracle Communications Cloud Native Core Network Slice Selection Function <release_number>* from **Release** drop-down.
6. Click **Search**. The **Patch Advanced Search Results** list appears.
7. Select the required ATS patch from the search results. The Patch Details window appears.
8. Click **Download**. The File Download window appears.
9. Click the **<p********_<release_number>_Tekelec>.zip** file to download the NSSF ATS package file.
10. Untar the zip file to get `ocats-nssf` directory, which consists of all the ATS Images. The `ocats-nssf` directory has the following files:

> **Note**
>
> **Prerequisites:**
>
> - To run oauth test cases for NSSF, oauth secrets needs to be generated. For more information, see "Configuring Secrets to Enable Access Token " section in *Oracle communications Cloud Native Core, Network Slice Selection Function Installation, Upgrade, Fault Recovery Guide*.

a. To ensure the functionality of the `Virtual_Host_NRF_Resolution_By_NSSF_Using_DNSSRV` ATS feature, the following configuration must be enabled as per the engineering team's guidance:

```
nrf-client:

  # This config map is for providing inputs to NRF-Client

  configmapApplicationConfig:

      enableVirtualNrfResolution=true

      virtualNrfFqdn=nrfstub.changeme-ocats.svc

      virtualNrfScheme=https
```

b. The necessary changes to the NSSF custom-values.yaml file are outlined below. For instance, if the NSSF ATS is deployed in the "`ocnssfats`" namespace, the `virtualNrfFqdn` configuration must be updated as follows:

```
nrf-client:
  # This config map is for providing inputs to NRF-Client
  configmapApplicationConfig:
      enableVirtualNrfResolution=true
      virtualNrfFqdn=nrfstub.ocnssfats.svc
      virtualNrfScheme=https
```

c. If the NSSF is installed with a HELM release name different from "`ocnssf`" (for example, the HELM release name is "`ocnssfats`"), the following parameter must be updated accordingly.
If the HELM release name is "`ocnssf`", no changes are required.

```
# Alternate Route Service Host Value
# Replace ocnssf with Release Name
alternateRouteServiceHost: ocnssf-alternate-route
```

d. If the HELM release name for NSSF is "`ocnssfats`", the following parameter must be updated accordingly.

```
# Alternate Route Service Host Value
# Replace ocnssf with Release Name
alternateRouteServiceHost: ocnssfats-alternate-route


ocats-nssf
├─── ocats-nssf-custom-configtemplates-25.2.200-README.txt   - file
contains all the information required for the package.
├─── ocats-nssf-custom-configtemplates-25.2.200.zip   - contains
serviceaccount,PVC File and Custom values file
├─── ocats-nssf-tools-pkg-25.2.200-README.txt   - file contains all
the information required for the package.
└─── ocats-nssf-tools-pkg-25.2.200.tgz   - file has the following
images and charts packaged as tar files
```

11. Untar the `ocats-nssf-tools-pkg-25.2.200.tgz` tar file

The structure of the file looks as given below:

```
ocats-nssf-tools-pkg-25.2.200
├─── amfstub-25.2.200.tar      - AMF Stub Server Docker image
├─── amfstub-25.2.200.tar.sha256
├─── ats_data-25.2.200.tar     - ATS data, After untar "ocnssf_tests"
folder will gets created in which ATS feature files present
├─── ats_data-25.2.200.tar.sha256
├─── ocats-nssf-25.2.200.tar   - NSSF ATS Docker Image
├─── ocats-nssf-25.2.200.tar.sha256
├─── ocats-nssf-25.2.200.tgz   - ATS Helm Charts, after untar "ocats-
nssf" ats charts folder gets created.
├─── ocats-nssf-25.2.200.tgz.sha256
├─── ocdns-bind-25.2.200.tar.  - NSSF DNS Stub Server Docker Image
├─── ocdns-bind-25.2.200.tar.sha256
└─── README.md
```

12. Copy the `ocats-nssf-tools-pkg-25.2.200.tgz` tar file to the CNE or Kubernetes cluster where you want to deploy ATS.
13. Along with the above packages, there is `ocats-nssf-custom-configtemplates-25.2.200.zip` at the same location.
    The readme file `ocats-nssf-custom-configtemplates-25.2.200-README.txt` contains information about the content of this zip file.

    Content of `ocats-nssf-custom-configtemplates-25.2.200.zip` is as below:

```
 Archive:  ocats-nssf-custom-configtemplates-25.2.200.zip
  inflating: nssf_ats_pvc_25.2.200.yaml
  inflating: ocats_nssf_custom_values_25.2.200.yaml
  inflating: ocats_ocnssf_custom_serviceaccount_25.2.200.yaml
```

    Copy these files to CNE or Kubernetes cluster where you want to deploy ATS.

```
ocats-nssf-tools-pkg-25.2.200
├─── amfstub-25.2.200.tar       - AMF Stub Server Docker image
├─── ats_data-25.2.200.tar      - ATS data, After untar "ocnssf_tests"
folder will gets created in which ATS feature files present
├─── ocats-nssf-25.2.200.tar    - NSSF ATS Docker Image
├─── ocats-nssf-25.2.200.tgz    - ATS Helm Charts, after untar "ocats-
nssf" ats charts folder gets created.
├─── ocdns-bind-25.2.200.tar.   - NSSF DNS Stub Server Docker Image
```

3.2.3 Deploying ATS in Kubernetes Cluster

To deploy ATS in Kubernetes Cluster:

1. Verify checksums of the tarballs mentioned in the file `Readme.txt`.
2. Run the following commands to extract tar file content, Helm charts, and Docker images of ATS:
   `tar -xvzf ocats-nssf-tools-pkg-25.2.200.tgz`

The output of this command will return the following files:

```
ocats-nssf-tools-pkg-25.2.200
├─── amfstub-25.2.200.tar      - AMF Stub Server Docker image
├─── amfstub-25.2.200.tar.sha256
├─── ats_data-25.2.200.tar     - ATS data, After untar "ocnssf_tests"
folder will gets created in which ATS feature files present
├─── ats_data-25.2.200.tar.sha256
├─── ocats-nssf-25.2.200.tar   - NSSF ATS Docker Image
├─── ocats-nssf-25.2.200.tar.sha256
├─── ocats-nssf-25.2.200.tgz   - ATS Helm Charts, after untar "ocats-
nssf" ats charts folder gets created.
├─── ocats-nssf-25.2.200.tgz.sha256
├─── ocdns-bind-25.2.200.tar.  - NSSF DNS Stub Server Docker Image
├─── ocdns-bind-25.2.200.tar.sha256
└─── README.md
```

3. **NSSF-ATS and Stub Images Load and Push**: Run the following commands in your cluster to load the `ocats` image and `amf stubserver` image:
   Docker Commands:

   ```
   docker load -i ocats-nssf-<version>.tar
   ```

   ```
   docker load -i amfstub-<version>.tar
   ```

   ```
   docker load -i ocdns-bind-<version>.tar
   ```

   Examples:

   ```
   docker load -i ocats-nssf-25.2.200.tar
   ```

   ```
   docker load -i amfstub-25.2.200.tar
   ```

   ```
   docker load -i ocdns-bind-25.2.200.tar
   ```

   Podman Commands:

   ```
   podman load -i ocats-nssf-<version>.tar
   ```

   ```
   podman load -i amfstub-<version>.tar
   ```

   ```
   podman load -i ocdns-bind-<version>.tar
   ```

Examples:

```
podman load -i ocats-nssf-25.2.200.tar


podman load -i amfstub-25.2.200.tar


podman load -i ocdns-bind-25.2.200.tar
```

4. Run the following commands to tag and push the ATS image registry.
   a. Run the following commands to grep the image:

      ```
      docker images | grep ocats-nssf


      docker images | grep amfstub


      docker images | grep ocdns
      ```

   b. Copy the Image ID from the output of the grep command and change the tag (version number) to your registry.
      Docker Commands:

      ```
      docker tag <Image_ID> <your-registry-name/ocats-nssf:<tag>>
      docker push <your-registry-name/ocats-nssf:<tag>>
      docker tag <Image_ID> <your-registry-name/amfstub:<tag>>
      docker push <your-registry-name/amfstub:<tag>>
      docker tag <Image_ID> <your-registry-name/ocdns-bind:<tag>>
      docker push <your-registry-name/ocdns-bind:<tag>>
      ```

      Podman Commands:

      ```
      podman tag <Image_ID> <your-registry-name/ats/ocats-nssf:<tag>>


      podman push <your-registry-name/ocats-nssf:<tag>>


      podman tag <Image_ID> <your-registry-name/amfstub:<tag>>


      podman push <your-registry-name/amfstub:<tag>>


      docker tag <Image_ID> <your-registry-name/ocdns-bind:<tag>>


      docker push <your-registry-name/ocdns-bind:<tag>>
      ```

5. **ATS Helm Charts** : Run the following command to get "ATS" Helm charts as shown below:

   ```
   tar -xvzf ocats-nssf-25.2.200.tgz
   ```

   The above command creates "ocats-nssf" helm charts of ATS.

6. **ATS Data**: Run the following command to get ATS data, which contains feature files and data:

   ```
   tar -xvf ats_data-25.2.200.tar
   ```

   The above command creates "ocnssf_tests" and "jobs" folders, which are copied inside after the ATS installation is complete.

   a. Copy the `ocnssf_tests` folder under NSSF ATS pod as shown below:

      ```
      kubectl cp ocnssf_tests <namespace>/<nssf ats podname>:/var/lib/jenkins/
      ```

      Example:

      ```
      kubectl cp ocnssf_tests cicdnssf-241204053638/ocats-nssf-8566d64cfb-cgmvc:/var/lib/jenkins/
      ```

   b. Copy the jobs folder under NSSF ATS pod as shown below:

      ```
      kubectl cp jobs <namespace>/<nssf ats podname>:/var/lib/jenkins/.jenkins/
      ```

      Example:

      ```
      kubectl cp jobs cicdnssf-241204053638/ocats-nssf-8566d64cfb-cgmvc:/var/lib/jenkins/.jenkins/
      ```

7. <Optional> Go to certificate folder inside ocats-nssf and run the following command:

   ```
   kubectl create secret generic ocnssf-secret --from-file=certificates/rsa_private_key_pkcs1.pem --from-file=certificates/trust.txt --from-file=certificates/key.txt --from-file=certificates/ocnssf.cer --from-file=certificates/caroot.cer -n ocnssf
   ```

8. **ATS Custom Values File Changes**: Update the image name and tag in the `ocats_nssf_custom_values_25.2.200.yaml` file as required.

   a. For this, open the `ocats_nssf_custom_values_25.2.200.yaml` file.

   b. Update the `image.repository` and `image.tag` parameters for `ocats-nssf`, `ocats-amf-stubserver`, `ocats-nrf-stubserver`, `ocats-nrf-stubserver1`, `ocats-nrf-stubserver2`, and `ocdns-bind`.

   c. Save and close the file after making the updates.

9. <Optional>**To enable static port**: ATS supports static port. By default, this feature is not available.

In the `ocats-nssf/values.yaml` file under service section, set the value of `staticNodePortEnabled` parameter as 'true' and provide a valid `nodePort` value for `staticNodePort`.

10. **ATS Service Account Creation**: In `ocats-nssf-custom-serviceaccount_.yaml`, change namespace as below:

    ```
    sed -i "s/changeme-ocats/${namespace}/g"
    ocats_ocnssf_custom_serviceaccount_.yaml
    ```

11. Run the following command to apply `ocats-nssf-custom-serviceaccount_.yaml` file:
    `kubectl apply -f <serviceaccount.yaml file> -n <namespace_name>`

    For example:

    ```
    kubectl apply -f ocats-nssf-custom-serviceaccount_.yaml -n ocnssf
    ```

12. **NSSF ATS PVC Creation — Ordered List**
    You can create the NSSF ATS PVC using either of the two available methods:
    a. Apply the NSSF ATS PVC YAML file manually after updating the namespace.
    b. Enable PVC creation directly from the NSSF ATS CV file.
    a. ***To create the PVC manually:***
        i. Update the namespace in the `nssf_ats_pvc_25.2.200.yaml` file using:

        ```
        sed -i "s/ocnssf/${namespace}/g" nssf_ats_pvc_25.2.200.yaml
        ```

        ii. Apply the updated PVC YAML using:

        ```
        kubectl apply -f <nssfatspvc.yaml file> -n <namespace_name>
        ```

        Example:

        ```
        kubectl apply -f nssf_ats_pvc_25.2.200.yaml
        ```

    b. ***To enable PVC creation in the NSSF ATS CV file***:
        i. Update the following parameters in the CV file:

        ```
          #The below section is related to the deployment of ATS with
        persistant volume
          #To enable PV set PVEnabled to true
          #To use an already existing pvc in the namespace, set the flag
        useExistingPVC to true, the name of it will be PVClaimName value.
          #retainPVC is to not delete the pvc even after un-installation of
        ATS. Make this true for this behavior, or leave it false for
        default behavior
          PVEnabled: false
          PVClaimName: "ocats-nssf-pvc"
          PVStorageClassName: "standard"
          PVStorage: "1Gi"
          RetainPVC: false
        ```

        When `PVEnabled` is set to `true`, a PVC is automatically created in the namespace using the name specified in `PVClaimName` (for example, `ocats-nssf-pvc`).

- PVC behavior after ATS uninstallation depends on the value of RetainPVC:
  - If `RetainPVC` is `false`, the PVC will be deleted when ATS is uninstalled.
  - If `RetainPVC` is `true`, the PVC will be retained even after ATS is uninstalled.

For example: `PVEnabled: true` and `RetainPVC: false`

- **Example 1**:

```
PVEnabled: true
PVClaimName: "<pvc_name>"
PVStorageClassName: "<storage_class>"
PVStorage: "<storage_size>"
RetainPVC: false
```

Before installation, no PVC exists:

```
kubectl get pvc -n <namespace>
NAME   STATUS   VOLUME   CAPACITY   ACCESS MODES
STORAGECLASS   AGE
```

Install ATS:

```
helm install <release_name> <chart_name> -n <namespace> -f
<ats_cv_file>
```

Check PVC after installation:

```
kubectl get pvc -n <namespace>
NAME             STATUS
VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS   AGE
<pvc_name>       Bound
<pvc_volume_id>                            1Gi
RWO            <storage_class>   <age>
```

Since `RetainPVC` is set to `false`, the PVC will be deleted after the uninstallation of ATS.

```
helm uninstall <release_name> -n <namespace>
release "<release_name>" uninstalled
```

After uninstallation, check the PVC status (PVC is deleted because `RetainPVC: false`):

```
kubectl get pvc -n <namespace>
NAME              STATUS        VOLUME              CAPACITY
ACCESS MODES   STORAGECLASS   AGE
<pvc_name>        Terminating   <pvc_volume_id>     1Gi
RWO            <storage_class>   <age>
```

- **Example 2**: `PVEnabled: true` and `RetainPVC: true`

  ```
  PVEnabled: true
  PVClaimName: "ocats-nssf-pvc"
  PVStorageClassName: "standard"
  PVStorage: "1Gi"
  RetainPVC: true
  ```

  Before installation, no PVC exists:

  ```
  kubectl get pvc -n <namespace>
  NAME   STATUS   VOLUME   CAPACITY   ACCESS MODES
  STORAGECLASS   AGE
  ```

  Install ATS:

  ```
  helm install <release_name> <chart_name> -n <namespace> -f
  <ats_cv_file>
  ```

  After installation, the PVC is created (because `PVEnabled: true`):

  ```
  kubectl get pvc -n <namespace>
  NAME                STATUS   VOLUME              CAPACITY
  ACCESS MODES   STORAGECLASS   AGE
  <pvc_name>          Bound    <pvc_volume_id>     1Gi
  RWO            <storage_class>   <age>
  ```

  Since `RetainPVC` is set to `true`, the PVC will be retained after the uninstallation of ATS.

  ```
  helm uninstall <release_name> -n <namespace>
  ```

  If `RetainPVC: true`, the PVC is not deleted and will be retained. The CLI output will indicate that the PVC is kept due to the resource policy. Check the PVC status after uninstallation:

  ```
  kubectl get pvc -n <namespace>
  NAME                STATUS   VOLUME              CAPACITY
  ACCESS MODES   STORAGECLASS   AGE
  <pvc_name>          Bound    <pvc_volume_id>     1Gi
  RWO            <storage_class>   <age>
  ```

13. **Pointing NSSF to Stub Servers**: Follow this step to point out NSSF to NRF-Stubserver and AMF-Stubserver in NSSF custom values file:

    ```
    sed -i "s/changeme-ocats/${namespace}/g" $NSSF_CUSTOM_DEPLOY_FILE
    ```

    For example:

    ```
    sed -i "s/changeme-ocats/${namespace}/g" ocnssf_custom_values_25.2.200.yaml
    ```

The NSSF custom values snippet is as follows:

```
nrf-client:
  # This config map is for providing inputs to NRF-Client
  configmapApplicationConfig:
    &configRef
    profile: |-
      [appcfg]
      primaryNrfApiRoot=nrf-stubserver.changeme-ocats:8080
      secondaryNrfApiRoot=nrf-stubserver.changeme-ocats:8080
# ARS Helm Configuration
staticVirtualFqdns:
    - name: https://abc.test.com
      alternateFqdns:
        - target: amf-stubserver.changeme-ocats
          port: 8080
          priority: 10
        - target: nrf-stubserver.changeme-ocats
          port: 8080
          priority: 20
    - name: http://xyz.test.com
      alternateFqdns:
        - target: amf-stubserver.changeme-ocats
          port: 8080
          priority: 10
        - target: nrf-stubserver.changeme-ocats
          port: 8080
          priority: 20
```

14. Deploy ATS as shown below. NSSF ATS Helm release name should be "ocats".
    `helm install <release_name> <charts> -n <namespace_name> -f <custom_values file> –-version <helm-chart-version>`

    For example:

    ```
    helm install ocats ocats-nssf -n ocnssf -f
    ocats_nssf_custom_values_25.2.200.yaml --version 25.2.200
    ```

    Running the above command creates the following pods:

    ```
    ocats-nssf

    ocats-amf-stubserver

    ocats-nrf-stubserver

    ocats-nrf-stubserver1

    ocats-nrf-stubserver2

    ocats-ocdns-bind
    ```

    For example:

    ```
    $ kubectl get pods -n cicdnssf-241204053638
    NAME                                                    READY
    STATUS      RESTARTS       AGE
    ```

```
ocats-amf-stubserver-5c5775dcb5-n9hlw                    2/2
Running     0              3h22m
ocats-nrf-stubserver-6bc6ff5ccc-6wkv7                    2/2
Running     0              3h22m
ocats-nrf-stubserver1-567f55488c-7bdxk                   2/2
Running     0              3h22m
ocats-nrf-stubserver2-697d4cffd9-c7dvr                   2/2
Running     0              3h22m
ocats-nssf-8566d64cfb-cgmvc                              2/2
Running     0              3h22m
ocats-ocdns-bind-967bd4bc8-hpcjn                         2/2
Running     0              3h22m
```

```
$ kubectl get svc -n cicdnssf-241204053638
NAME                                    TYPE           CLUSTER-IP
EXTERNAL-IP   PORT(S)             AGE
amf-stubserver                          LoadBalancer   10.96.27.125
<pending>     8080:31064/TCP      5h34m
dnssim                                  ClusterIP      10.96.99.119
<none>        53/UDP,6236/TCP     5h34m
nrf-stubserver                          LoadBalancer   10.96.192.123
<pending>     8080:30409/TCP      5h34m
nrf-stubserver1                         LoadBalancer   10.96.235.187
<pending>     8080:31157/TCP      5h34m
nrf-stubserver2                         LoadBalancer   10.96.77.69
<pending>     8080:31056/TCP      5h34m
ocats-nssf                              LoadBalancer   10.96.14.30
<pending>     8080:31995/TCP      5h34m
```

15. Run the following command to verify the ATS deployment:
    `helm status <release_name>`

    The following screenshot is an example of a successful ATS deployment, where `STATUS:DEPLOYED` is an indicator of the same.

```
$ helm status ocats -n cicdnssf-241204053638
NAME: ocats
LAST DEPLOYED: Wed Dec 4 05:37:54 2024
NAMESPACE: <namespace>
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Virtual_Host_NRF_Resolution_By_NSSF_Using_DNSSRV ATS Feature

To ensure the functionality of the **Virtual_Host_NRF_Resolution_By_NSSF_Using_DNSSRV** ATS feature, the following configuration must be enabled:

```
nrf-client:
  # Configuration map for providing inputs to the NRF-Client
  configmapApplicationConfig:
    enableVirtualNrfResolution: true
```

```
    virtualNrfFqdn: nrfstub.changeme-ocats.svc
    virtualNrfScheme: http
```

The necessary changes to the **NSSF** `custom-values.yaml` file are outlined below:

Example Configuration for Namespace "ocnssfats"

If the NSSF ATS is deployed in the `ocnssfats` namespace, update the `virtualNrfFqdn` configuration as follows:

```
nrf-client:
  # Configuration map for providing inputs to the NRF-Client
  configmapApplicationConfig:
    enableVirtualNrfResolution: true
    virtualNrfFqdn: nrfstub.ocnssfats.svc
    virtualNrfScheme: http
```

Configuration for Different HELM Release Names

- **Default Release Name** (`ocnssf`): No changes are required.
- **Alternate Release Name**: If the HELM release name is different (e.g., `ocnssfats`), update the following parameter:

  ```
  # Alternate Route Service Host Value
  # Replace 'ocnssf' with the HELM release name
  alternateRouteServiceHost: ocnssfats-alternate-route
  ```

Steps to Integrate DNS Stub with NSSF

1. Enable virtual FQDN resolution in the nrf-client management app profile configuration. Example:

   ```
   nrf-client:
       profile: |-
         [appcfg]
         primaryNrfApiRoot=nrf-stubserver.changeme-ocats:8080
         secondaryNrfApiRoot=nrf-stubserver.changeme-ocats:8080
         nrfScheme=http
         retryAfterTime=PT120S
         nrfClientType=NSSF
         nrfClientSubscribeTypes=
         appProfiles=[{"nfInstanceId":"9faf1bbc-6e4a-4454-a507-
   aef01a101a01","nfType":"NSSF","nfStatus":"REGISTERED","heartBeatTimer":30,"
   fqdn":"ocnssf-
   nsgateway.ocnssf.svc","priority":1,"capacity":1,"load":2,"plmnList":
   [{"mcc":"311","mnc":"480"}],"nfSetIdList":
   ["setEast.nssfset.5gc.mnc480.mcc311"],"locality":"rcnltxekloc1","nfServices
   ":[{"serviceInstanceId":"92d59bfc-e5d6-47f5-
   a26b-3a03facdebcc","serviceName":"nnssf-nsselection","versions":
   [{"expiry":null,"apiFullVersion":"1.0.0","apiVersionInUri":"v1"}],"scheme":
   "http","nfServiceStatus":"REGISTERED","fqdn":"ocnssf1-ingress-
   gateway.ocnssf.svc","interPlmnFqdn":null,"ipEndPoints":
   [{"ipv4Address":"10.224.45.178","transport":"TCP","port":80}],"allowedNfTyp
   es":["AMF","NSSF"],"priority":1,"capacity":1,"load":2},
   {"serviceInstanceId":"d33728cd-6e21-434b-bc5a-
   ```

```
ed69bc612377","serviceName":"nnssf-nssaiavailability","versions":
[{"expiry":null,"apiFullVersion":"1.0.0","apiVersionInUri":"v1"}],"scheme":
"http","nfServiceStatus":"REGISTERED","fqdn":"ocnssf2-ingress-
gateway.ocnssf.svc","interPlmnFqdn":null,"ipEndPoints":
[{"ipv4Address":"10.224.45.179","transport":"TCP","port":80}],"allowedNfTyp
es":["AMF","NSSF"],"priority":1,"capacity":1,"load":2}]}]
      enableF3=true
      enableF5=true
      renewalTimeBeforeExpiry=3600
      validityTime=30
      enableSubscriptionAutoRenewal=true
      nfHeartbeatRate=80
      acceptAdditionalAttributes=false
      retryForCongestion=5
      enableVirtualNrfResolution=true
      virtualNrfFqdn=nrfstub.changeme-ocats.svc
      virtualNrfScheme=https
```

2. In the above configuration, since "`enableVirtualNrfResolution=true`" and "`virtualNrfFqdn=nrfstub.changeme-ocats.svc`" have been set, the NRF client management attempts to contact the Egress Gateway, and Egress Gateway attempts to connect Alternate Route Service for virtual FQDN Resolution. In this service, we configure static settings related to the FQDN, as shown below:

```
#Static virtual FQDN Config
staticVirtualFqdns:
  - name: https://abc.test.com
    alternateFqdns:
      - target: amf-stubserver.changeme-ocats
        port: 8080
        priority: 10
      - target: nrf-stubserver.changeme-ocats
        port: 8080
        priority: 20
  - name: http://xyz.test.com
    alternateFqdns:
      - target: amf-stubserver.changeme-ocats
        port: 8080
        priority: 10
      - target: nrf-stubserver.changeme-ocats
        port: 8080
        priority: 20
```

According to the static virtual FQDN configuration under ARS, there is no record for the FQDN "`virtualNrfFqdn=nrfstub.changeme-ocats.svc`". As a result, ARS will attempt to resolve this FQDN by contacting the default CoreDNS server (the default Kubernetes cluster DNS server). To ensure proper resolution, we need to point ARS to the DNS stub we installed by editing the ARS deployment file after deploying NSSF.

3. Before running ATS Test suite, nrf-client-nfdiscovery and nrf-client-nfmanagement pods has to be restarted. Here are the steps to edit the "alternate route service" deployment pointing towards DNS Stub:

   a. Run the follwing command to get the list of pods:

```
$ kubectl get pods -n <namespace>
```

For example:

```
$ kubectl get pods -n ocnssf
```

Expected sample response:

```
NAME                                                    READY
STATUS      RESTARTS      AGE
ocats-amf-stubserver-6f57f7f57f-bk75w                   2/2
Running     0             64m
ocats-nrf-stubserver-5f89cdb74c-649kx                   2/2
Running     0             64m
ocats-nssf-76cf4fc678-vrl24                             2/2
Running     0             64m
ocdns-bind-5975fc59c8-6vllx                             2/2
Running     0             41m
ocnssf-alternate-route-5f857d94bb-dm5vl                 2/2
Running     0
40m                                          0/1     Completed
0             30m
```

b. Run following command to get searches information from dns-bind pod to enable communication between Alternate Route and dns-bind service:

```
kubectl exec -it <dnsbind-pod> -n <NAMESPACE> -- /bin/bash -c 'cat /etc/
resolv.conf' | grep search | tr ' ' '\n' | grep -v 'search'
```

For example:

```
$ kubectl exec -it ocdns-bind-5975fc59c8-6vllx -n ocnssf -- /bin/bash -
c 'cat /etc/resolv.conf' | grep search | tr ' ' '\n' | grep -v 'search'
```

Expected sample response:

```
ocnssf.svc.cluster.local
svc.cluster.local
cluster.local
gbucdsint02phx.oraclevcn.com
snphxprshared1.gbucdsint02phx.oraclevcn.com
```

c. By default alternate service will point to CoreDNS and you will see following settings in deployment file. Now, edit alternate route service deployment and add configuration as below:

```
$ kubectl edit deployment ocnssf-alternate-route -n <NAMESPACE>
```

For example:

```
$ kubectl edit deployment ocnssf-alternate-route -n ocnssf


terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
dnsPolicy: ClusterFirst
restartPolicy: Always
```

d. Edit the deployment file to add following content in alternate service to query DNS stub:

- Add the nameservers IPaddress which you recorded after installing the DNS stub (cluster IP of DNS Stub)
- Add all the search information one by one which you recorded earlier.
- Set dnsPolicy to "None"

```
dnsConfig:
  nameservers:
  - <dns_stub_cluster_ip_address>
  searches:
  - dns-bind search
  - dns-bind search
  - dns-bind search
  dnsPolicy: None


$ kubectl get svc -n ocnssf
```

Expected sample response:

```
NAME                                      TYPE           CLUSTER-IP
EXTERNAL-IP   PORT(S)             AGE
dnssim                                    ClusterIP      10.96.65.56
<none>        53/UDP,6236/TCP     48m
```

For example:

```
terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
dnsConfig:
  nameservers:
  - 10.96.65.56
  searches:
  - ocnssf.svc.cluster.local
  - svc.cluster.local
  - cluster.local
  - gbucdsint02phx.oraclevcn.com
  - snphxprshared1.gbucdsint02phx.oraclevcn.com
dnsPolicy: None
restartPolicy: Always
```

Verification

Use the following `curl` command to verify the setup:

```
curl -v --http2-prior-knowledge -X GET "http://ocnssf-alternate-route:80/
lookup?fqdn=nf1stub.ocnssf.svc&scheme=h
ttp"
```

Expected sample response:

```
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 10.96.194.219...
* TCP_NODELAY set
* Connected to ocnssf-alternate-route (10.96.194.219) port 80 (#0)
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade:
len=0
* Using Stream ID: 1 (easy handle 0x55b19b19c6f0)
> GET /lookup?fqdn=nf1stub.ocnssf.svc&scheme=http HTTP/2
> Host: ocnssf-alternate-route
> User-Agent: curl/7.61.1
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS == 2147483647)!
< HTTP/2 200
< content-type: application/json
< date: Tue, 08 Oct 2024 05:48:02 GMT
< x-envoy-upstream-service-time: 5
<
* Connection #0 to host ocnssf-alternate-route left intact
[{"target":"ocats-amf-
stubserver.ocnssf.svc","port":8080,"ttl":86400,"type":"SRV","dclass":"IN","pri
ority":1,"weight":60},{"target":"ocats-nrf-
stubserver.ocnssf.svc","port":8080,"ttl":86400,"type":"SRV","dclass":"IN","pri
ority":20,"weight":20},{"target":"ocats-nrf-
stubserver1.ocnssf.svc","port":8080,"ttl":86400,"type":"SRV","dclass":"IN","pr
iority":30,"weight":20}][ocnrfusr@ocnssf-ocnssf-nrf-client-
nfmanagement-744ff58956-tfr5g app]$
```

4
Running NF Test Cases Using ATS

This section describes how to run NF test cases using ATS. It includes:

- Running BSF Test Cases using ATS
- Running NSSF Test Cases using ATS

4.1 Running BSF Test Cases using ATS

This section describes how to run BSF test cases using ATS.

> **Note**
>
> Restart the NRF-client pod of BSF for UDR and CHF discovery as part of each test case.

4.1.1 Prerequisites

To run Binding Support Function test cases, ensure that the following prerequisites are met:

- Deploy BSF 25.2.200 with default helm configurations using helm charts to run all test cases. The ATS version must be compatible with BSF 25.2.200.
  For more information on how to install BSF, see *Oracle Communications Cloud Native Core, Binding Support Function Installation, Upgrade, and Fault Recovery Guide*.
- Go-STUB must be installed in the same namespace where ocbsf is installed.
- Add the following to Kubernetes namespace to grant role access:

```
PolicyRule:
  Resources              Non-Resource URLs  Resource Names  Verbs
  ---------              -----------------  --------------  -----
  pods/log               []                 []              [get list]
  configmaps             []                 []              [watch get
list delete update create]
  pods                   []                 []              [watch get
list delete update create]
  secrets                []                 []              [watch get
list delete update create]
  services               []                 []              [watch get
list delete update create]
  deployments.apps       []                 []              [watch get
list update]
  replicasets.apps       []                 []              [watch get
list update]
  deployments.extensions  []                []              [watch get
list update]
```

```
  replicasets.extensions  []                  []              [watch get
list update]
```

- ATS Prometheus metrics validation works only when:
  - the metrics suffixes are not configured
  - installation has a single pod for each microservice in the BSF deployment
- You can customize test cases in the custom test case folders (cust_newfeatures, cust_regression and cust_performance). You can add new test cases, remove unwanted test cases and modify existing test cases. It does not impact the original product packaged test cases available in the newfeatures, regression and performance folders respectively. For more details about custom test case folders, see Custom Folder Implementation.
- Install Prometheus server in the cluster.
- Database cluster is in the running state with all required tables. Verify that there are no previous entries in the database before running test cases.
- Do not initiate a job in two different pipelines at the same time.
- For running ATS features, ensure to update the following mandatory parameters in ocbsf_custom_values_25.2.200.yaml file only when you are not using the minimal custom values.yaml file.

```
logging:
  burst:
    rate: 500
    max: 3000
    onMismatch: DENY
    logLevel: DEBUG
```

> **Note**
>
> Please ensure that you use the latest version of the Custom Values file when installing BSF initially.

- For using Controlled Shutdown feature, ensure that `enableControlledShutdown` parameter is enabled on BSF during installation. If this parameter is not enabled, the test case for this feature will fail.
- In the `application-config` configmap, configure the following parameters with the respective values:
  - `primaryNrfApiRoot=nf1stub.<namespace_gostubs_are_deployed_in>.svc:8080`

    For example:

    `primaryNrfApiRoot=nf1stub.ocats.svc:8080`
  - `secondaryNrfApiRoot=nf11stub.<namespace_gostubs_are_deployed_in>.svc:8080`

    For example:

    `secondaryNrfApiRoot=nf11stub.ocats.svc:8080`
  - `virtualNrfFqdn = nf1stub.<namespace_gostubs_are_deployed_in>.svc`

    For example:

    `virtualNrfFqdn=nf1stub.ocats.svc`

- `retryAfterTime=PT30S`

- To enable ATS BSF GUI with the HTTPS protocol, the above mentioned `application-config` configmap related following parameters should be configured with the respective values:

```
# Please edit the object below. Lines beginning with '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  profile: | -
    [appcfg]
    primaryNrfApiRoot=nf1stub.ocats.svc:8443
    secondaryNrfApiRoot=nf11stub.ocats.svc:8443
    nrfScheme=https
    virtualNrfPort=8443
    virtualNrfScheme=https
```

- Before running ATS Test suite, restart `nrf-client-nfdiscovery` and `nrf-client-nfmanagement` pods.
  - Run the following command to get all the configmaps in your namespace.

    ```
    kubectl get configmaps -n <BSF_namespace>
    ```

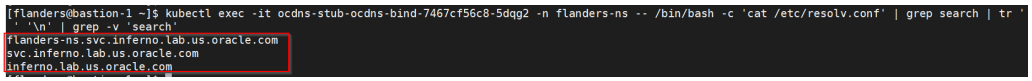
  - Edit the `alternate route service` deployment pointing towards DNS Stub.

    Run the following command to get searches information from `dns-bind` pod to enable communication between Alternate Route and `dns-bind` service.

    ```
    kubectl exec -it <dns-bind pod> -n <NAMESPACE> -- /bin/bash -c
    'cat /etc/resolv.conf' | grep search | tr ' ' '\n' | grep -v 'search'
    ```

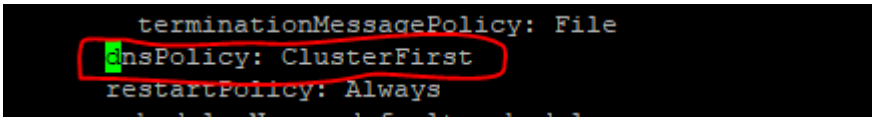
    Example:

Figure 4-1 Editing Alternate Route Service deployment pointing towards DNS Stub



    By default, Alternate Route Service points to CoreDNS.

Figure 4-2 Alternate Route Service settings in deployment file



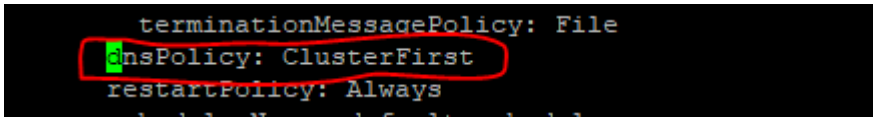
Change the deployment file to add following content in alternate service to query DNS stub.

```
kubectl edit deployment ocbsf-occnp-alternate-route -n ocbsf
```

* Add the nameservers IPaddress.
* Add all the search information.
* Set dnsPolicy to `None`.

```
dnsConfig:
  nameservers:
   - <dns_stub_cluster_ip_address>
   searches:
   - dns-bind search
   - dns-bind search
   - dns-bind search
   dnsPolicy: None
```

Figure 4-3 dnsConfig



– If Service Mesh check is enabled, create a destination rule to fetch the metrics from the Prometheus. For destination rule to communicate between TLS enabled entity (ATS) and non-TLS entity (Prometheus), Prometheus is kept outside of the service mesh. The rule can be created as follows:

```
kubectl apply -f - <<EOF

apiVersion:networking.istio.io/v1alpha3
kind:DestinationRule
metadata:
  name:prometheus-dr
  namespace:ocats
spec:
  host:oso-prometheus-server.ocbsf.svc.cluster.local
  trafficPolicy:
    tls:
      mode:DISABLE
EOF
```

* `name`: indicates the name of destination rule.
* `namespace`: indicates where the ATS is deployed.
* `host`: indicates the hostname of the prometheus server.

- For Bsf_To_Nrf_Late_Arrival feature file to be executed, nfInstanceId should be configured in Bsf_To_Nrf_Late_Arrival.yaml parameterization file. This nfInstanceId must be the same as the one under BSF nfInstanceId, that is configured during BSF installation. The

nfInstanceId must to be configured in ATS UI under pipeline configuration (BSF_NFINSTANCE_ID).

- To ensure consistent functioning of ATS related to the audit service, modify the Audit deployment by reducing the value of AUDIT_NOTIFY_SCHEDULER_POLLING_INTERVAL_MILLISEC from 30000 to 1000 (changing from 30 seconds to 1 second).

```
$ kubectl get deploy -n ocbsf | grep 'audit' ocbsf-ocpm-audit-service
1/1     1      1     10h


$ kubectl edit deploy ocbsf-ocpm-audit-service -n ocbsf
```

The deployment will open. Scroll down until below fields are seen

```
- name:  AUDIT_NOTIFY_SCHEDULER_POLLING_INTERVAL_MILLISEC value: "30000"
```

Update the value of AUDIT_NOTIFY_SCHEDULER_POLLING_INTERVAL_MILLISEC to 1000 if it isn't already.

Application Config map changes for BSF registrations over TLS

To enable TLS communication make the following changes by editing the config map of application-config:

- NRF port from 8080 to 8443
- nrfScheme to https

Sample:

```
apiVersion: v1
data:
  profile: |-
    [appcfg]
    primaryNrfApiRoot=nf1stub.ocats.svc:8443
    secondaryNrfApiRoot=nf11stub.ocats.svc:8443
    nrfScheme=https
    virtualNrfPort=8443
    virtualNrfScheme=https
```

> **Note**
>
> In the config map of application-config, delete the lines which has *supportedDataSetId* or *secondaryNrfApiRoot* strings.

4.1.2 Logging into ATS

Before logging into ATS GUI, it is important to get the worker node external IP and node port of the service, 'ocats-bsf'.

Run the following command to get the external IP for the worker node:

Example:

```
kubectl get nodes -owide
```

The output of the command is:

```
ocbsf-k8s-node-1        Ready    <none>   111d   v1.16.7   192.168.200.26
10.75.152.111   Oracle Linux Server 7.8
4.14.35-1902.303.5.3.el7uek.x86_64   containerd://1.2.10
```

Run the following command to get the nodeport:

```
kubectl get svc -n <BSF_namespace>
```

Example:

```
kubectl get svc -n ocbsf
```

The output of the command is:

```
ocbsf-ocats-ocats-bsf     LoadBalancer   10.233.53.144   10.75.225.49
8080:31944/TCP      19h
```

To log in to ATS, open the web browser and type the following URL:

http://<Worker-Node-IP>:<Node-Port-of-ATS>

If the 'ocats-bsf' Service has an external IP available, <SVC external IP> can also be used to log in to ATS.

```
http://<External IP of ATS Service>:8080
```

Example:

```
http://10.75.225.49:8080
```

Running ATS

To run ATS test cases, perform the following steps:

1. Enter the **username** as `bsfuser` and **Password** as `bsfpasswd`.
2. Click **Sign in**.

> **Note**
>
> To modify default login password, see Modifying Login Password.

Figure 4-4 Pre-configured Pipelines

On successful log in, users should see the following pipelines:

- **BSF-NewFeatures:** This pipeline has all the new test cases delivered for BSF 25.2.200.
- **BSF-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
- **BSF-HealthCheck:** This pipeline checks if BSF and ATS are deployed correctly. This shows only when the user has enabled this feature at the time of installing BSF ATS.
- **BSF-Regression:** This pipeline has all the test cases delivered in BSF ATS - 25.2.100.

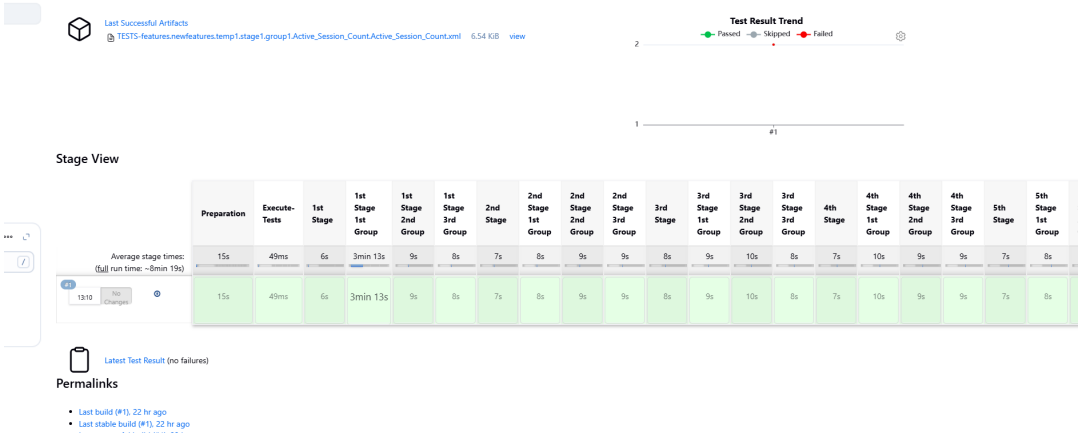
4.1.3 Running BSF_NewFeatures Pipeline

BSF_NewFeatures Pipeline

This is a pre-configured pipeline where users can run all the BSF new test cases. To configure its parameters, which is a one time activity, perform the following steps:

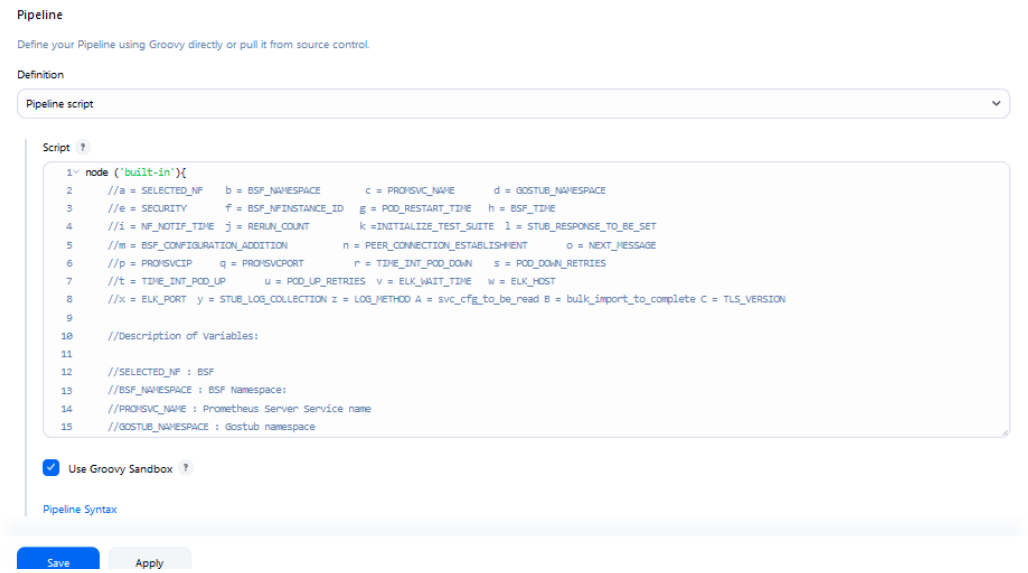
1. Click **BSF_NewFeatures** in the **Name** column and then, click **Configure** in the left navigation pane as shown below:

Figure 4-5 BSF New Feature Pipeline



2. The BSF_NewFeatures, **General** tab appears. Make sure that the screen loads completely.
3. Scroll-down to the end. The control moves from **General** tab to the **Pipeline** tab as shown below:

Figure 4-6 BSF New Features Pipeline Configuration



You can change the parameter values from **"a"** - to - **"x"** as per user requirement. The parameter details are available as comments from line number **2** - to - **7**. In the **Script** area of the Pipeline section, you can change the values of the following parameters:

- **a:** Name of the NF to be tested in capital (BSF)
- **b:** Change this parameter to update the namespace where BSF was deployed in your bastion.
- **c:** Name of Prometheus service namespace (occne-prometheus-server)
- **d:** Change this parameter to update the namespace where your gostubs are deployed in your bastion.
- **e:** Set this parameter as 'unsecure', if you intend to run ATS in TLS disabled mode. Else, set this parameter as 'secure'.
- **f:** Configure this parameter to set BSF_NFINSTANCE_ID.
- **g:** Set a value more than 45 seconds for this parameter. The default wait time for the pod to come up is 45 seconds. Every TC requires restart of the nrf-client-management pod.
- **h:** Set a value more than 60 seconds for this parameter. The default wait time to add a configurations to the database is 60 secs.
- **i:** Set this parameter to more than 140 secs. The default wait time for Nf_Notification Test Cases is given as 140 secs.
- **k:** Use this parameter to set the waiting time to initialize Test Suite.
- **l:** Use this parameter to set the waiting time to get response from Stub.

- **m:** Use this parameter to set the waiting time after adding BSF Configuration.
- **n:** Use this parameter to set the waiting time for Peer connection establishment.
- **o:** Use this parameter to set the waiting time before sending next message.
- **p:** Use this parameter to set Prometheus Server IP.
- **q:** Use this parameter to set Prometheus Server Port.

> **Note**
>
> If **PrometheusAuthEnabled** is set to **true** during ATS installation, then set **p** and **q** accordingly.

- **r:** Use this parameter to set the interval after which the POD status is checked when it is down.
- **s:** Use this parameter to set number of retry attempt in which will check the pod down status
- **t:** Use this parameter to set the interval after which we check the POD status if its UP.
- **u:** Use this parameter to set number of retry attempt in which will check the pod up status.
- **v:** Use this parameter to set Wait time to connect to Elastic Search.
- **w:** Use this parameter to set Elastic Search HostName.
- **x:** Use this parameter to set Elastic Search Port.
- **y:** Use this parameter to set To Enable/Disable Stub logs collection.
- **z:** Use this parameter to set Log collection endpoint either Elasticsearch or Kubernetes.
- **A:** Use this parameter to set Timer to wait for importing service configurations.
- **B:** Use bulk_import_to_complete to add custom time in Jenkins post bulk imports.
- **C:** Use this parameter to set TLS version (1.2 or 1.3). The default value is 1.2.

(*Optional*)To collect application logs per failed scenario, user can configure the values for the following parameters:

- **z:** If you want log collection to happen through Elastic search, set the value for this parameter as `Elasticsearch`. If not, specify the value as `Kubernetes`.
  If you want to collect logs through Elastic search, it is required to configure the values for the following parameters:
  - **v:** Specifies the wait time to connect to Elastic search (`ELK_WAIT_TIME`).
  - **w:** Specifies the host name of Elastic search (`ELK_HOST`). For example, `occne-elastic-elasticsearch-master.occne-infra/`
  - **x:** Specifies the port for Elastic search (`ELK_PORT`). For example, 9200.
- **y:** If you want to collect stub logs, set the value for this parameter as `yes`. If not, specify the value as `no`.

4. Click **Save** after updating the parameter values. The **BSF_NewFeatures** Pipeline page appears.

> **Note**
>
> It is recommended to save a copy of the pipeline script in your local machine that you may refer while restarting ATS pods.

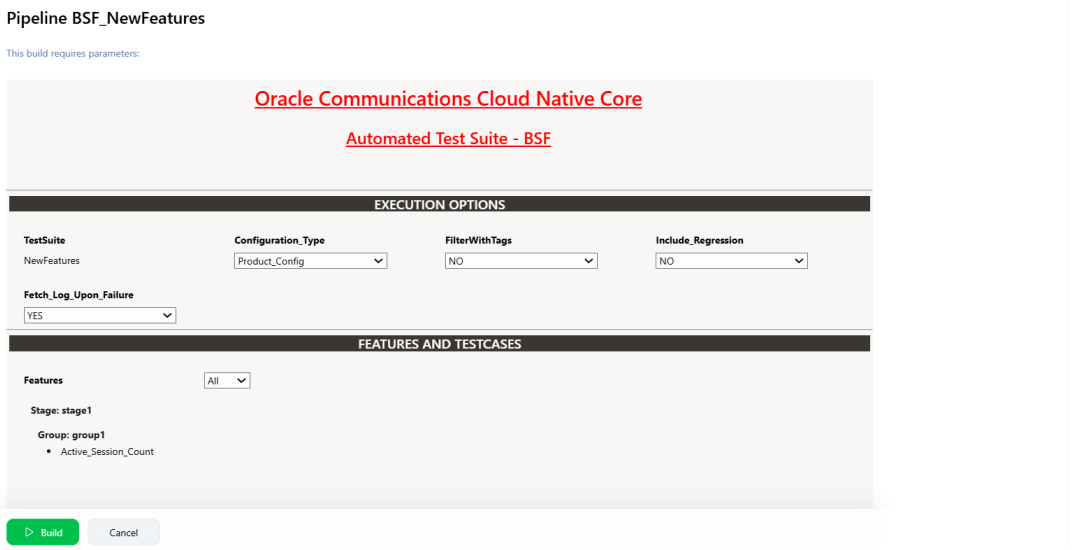
> **NOT_SUPPORTED**
>
> Do not modify anything other than the parameter values described in this section.

Running BSF Test Cases

To run BSF test cases, perform the following steps:

1. Click the **Build with Parameters** link available in the left navigation pane of the **BSF_NewFeatures** Pipeline screen. The following page appears.

Figure 4-7 BSF New Features Build with Parameters



> **Note**
>
> Make sure that the value of **FilterWithTags** and **Include_NewFeatures** is selected as **NO**.

2. If you want to collect logs for any given build, select YES from the drop-down menu of **Fetch_Log_Upon_Failure**.
3. Click **Build** and select **Console Output** to view the test results. The following is a sample test result ouput:

Figure 4-8 Sample: Test Result Output in Console

> **Note**
>
> For more details on consolidated test report, see Managing Final Summary Report, Build Color, and Application Log.

Queuing Jenkins Jobs

Using this feature, you can queue a second job even when current job is still running. The second job can be triggered either from the same or a different pipeline.

The following table lists various scenarios on how queuing works:

Table 4-1 Queuing Jenkins Jobs

Concurrent Builds	New Features Current Build	New Features New Build	Regression Current Build	Regression New Build Build	Result
Enabled	Running	Triggered	NA	NA	New-Build of New-Features is added to queue.
Enabled	Running	NA	NA	Triggered	New-Build of Regression is added to queue.
Disabled	NA	NA	Running	Triggered	New-Build of Regression is added to queue.
Disabled	NA	Triggered	Running	NA	New-Build of New-Features is added to queue.

Extracting Application Logs

If any scenarios fail after running the pipeline job, perform the following steps:

1. Log in to the ATS pod:

   ```
   kubectl exec -it pod/ocats-bsf-6f6dfc76b5-jbgzt -n ocbsf bash
   ```

2. Go to Jenkins build directory:

   ```
   cd $JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/
   ```

   For example:

   ```
   cd /var/lib/jenkins/.jenkins/jobs/BSF_Regression/builds/2
   ```

3. Extract the `applogs.zip` file:

   ```
   unzip applogs.zip
   ```

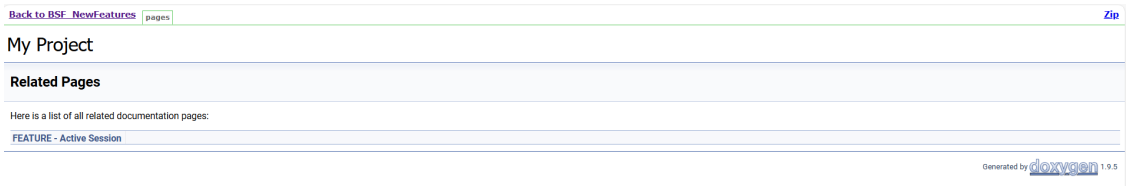
4. After successfully unzipping the file, open the `applog` folder to view the pod logs for failed scenarios:

   ```
   (env) [jenkins@ocats-bsf-6f6dfc76b5-jbgzt applog]$ pwd
   /var/lib/jenkins/.jenkins/jobs/BSF_Regression/builds/2/applog
   (env) [jenkins@ocats-bsf-6f6dfc76b5-jbgzt applog]$ ls -ltrh
   total 760K
   -rw-r--r--. 1 jenkins jenkins 250K Nov 19 11:08 Initial_Run-
   Register_BSF_With_NFSetIDList.log
   -rw-r--r--. 1 jenkins jenkins 249K Nov 19 11:08 1st_Rerun-
   Register_BSF_With_NFSetIDList.log
   -rw-r--r--. 1 jenkins jenkins 255K Nov 19 11:09 2nd_Rerun-
   Register_BSF_With_NFSetIDList.log
   ```

4.1.4 BSF_NewFeatures Documentation

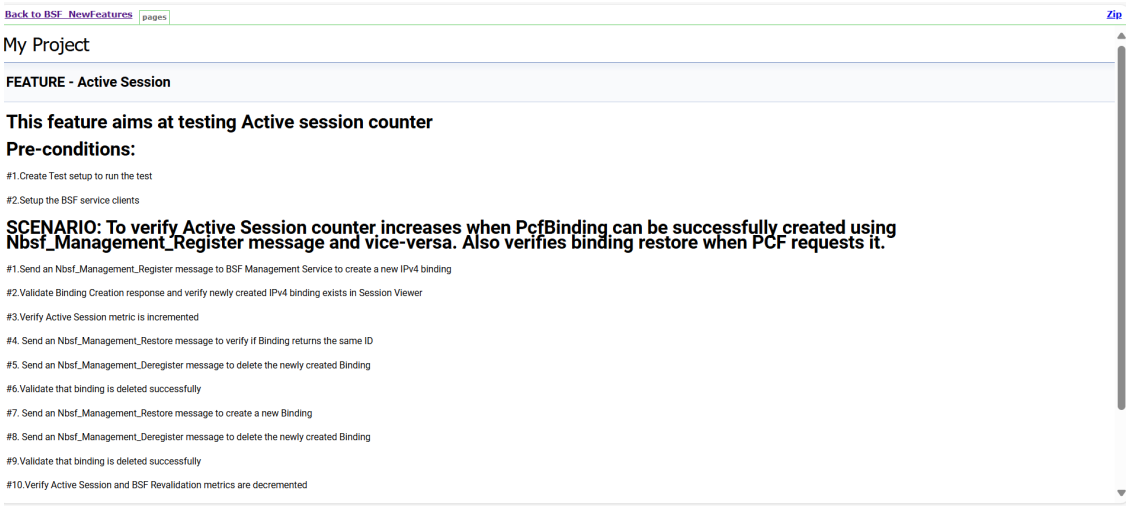
To view BSF functionalities, go to **BSF_NewFeatures** pipeline and click **Documentation** link in the left navigation pane.

Figure 4-9 BSF_NewFeatures Feature List Documentation



Click any functionality to view its test cases and scenarios of each test case. For example, when you click **FEATURE - BSF_Error_Response_Enhancements**, the following test description appears:

Figure 4-10 Test Cases and Scenarios of Feature - BSF_Error_Response_Enhancements



Based on the functionalities covered under **Documentation**, the **Build Requires Parameters** screen displays test cases. To navigate back to the pipeline BSF-NewFeatures screen, click **Back to BSF_NewFeatures** link available on top left corner of the screen.

Test Result Analyzer

Using the Test Result Analyzer plug-in available in ATS, user can view consolidated and detailed reports. For more information, see Test Results Analyzer section.

Test Case Mapping to Features and Display Total Counts

With this feature, users can view Total Count of Features, TestCases/Scenarios and TestCase mapping to each Feature of BSF in ATS View. For more information, see Support for Test Case Mapping and Count section.

Stub Predefined_priming Support

Stub Predefined_priming configuration in ATS enables ATS to respond back with the payload message instead of default message when prime configuration does not match with the feature level priming or when the sub is not primed.

When the Predefined_priming is not configured and when a request is received at the stub that mimics NRF:

1. Stub checks against the feature level prime configuration.
2. If a match is found in feature level prime, stub replies with the payload message.
3. If there is no match found, stub replies with the default response.

```
stub_log: No match found in prime configuration for sending the response.
Sending default - 200
ATS log: {default_response}
```

When the Predefined_priming is configured and when a request is received at the stub that mimics NRF:

1. Stub checks against the feature level prime configuration.

2. If a match is found in feature level prime, feature level prime is used for responding to the request.
3. If a match is not found in the feature level prime, but found in the pre-configured prime, pre-configured prime is used for responding to the request.
4. If a match is found in both pre-primed as well as feature level prime, feature level prime configuration is given priority and the same is used for responding to the request.
5. If a match is not found in both pre-primed as well as feature level prime, stub sends a default response.

4.1.5 Running BSF Regression Pipeline

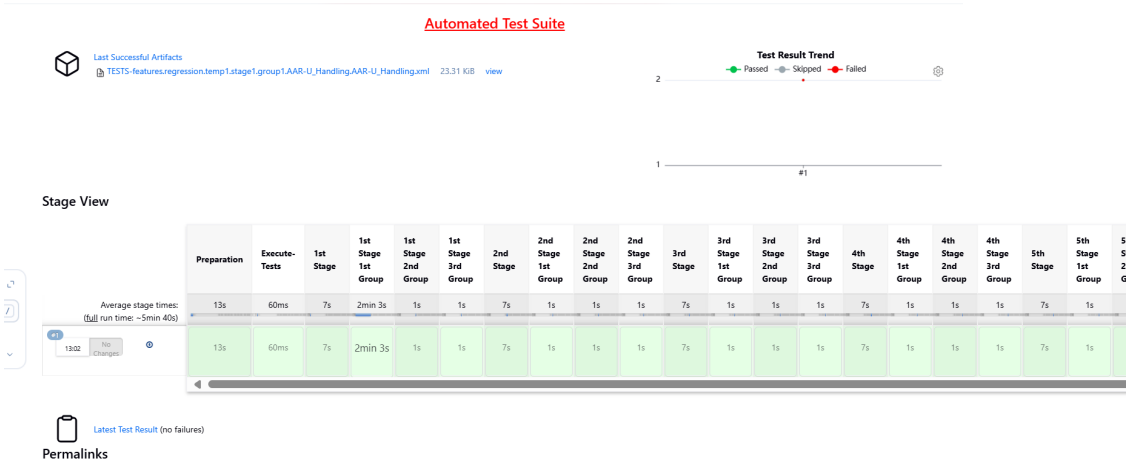
This section describes how to run test cases for Binding Support Function (BSF) Regression pipeline.

The **BSF_Regression** pipeline is a pre-configured pipeline where all the test cases of previous releases are available. For example, for BSF 25.2.200, this pipeline has all the test cases released till BSF 25.2.100.

To view **BSF_Regression** pipeline details, perform the following steps:

1. Click **BSF_Regression** in the **Name** column.
2. Click **Build with Parameters** in the left navigation pane.
3. Copy the required test cases that are available in the BSF folder and place them appropriately within the **custom** folder for BSF_Regression.
4. Reload the page to view the test cases available in the **custom** Regression folder.
5. Click **Build**.

Figure 4-11 BSF_Regression Pipeline



> **Note**
>
> The Regression pipeline does not have any sanity option. However, users must perform all the steps performed in the **BSF_NewFeatures** pipeline. Ensure that the pipeline script is configured according to the environment variables.

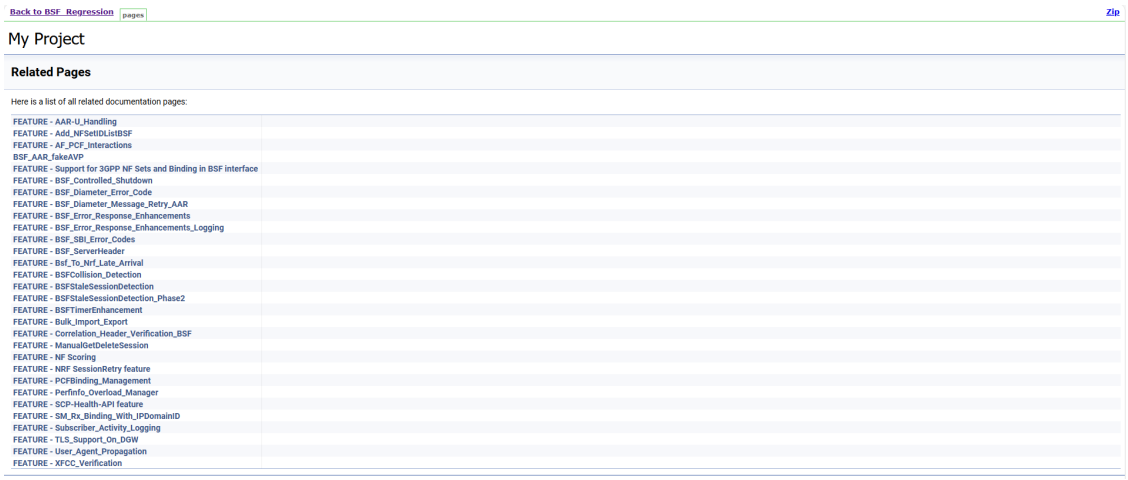
4.1.6 BSF_Regression Documentation

This section describes the documentation for **BSF_Regression** pipeline.

To view the documentation for any of the BSF features, on the ATS home page, click BSF_Regression. Then, click **Documentation** in the left navigation pane.

This page shows features of only those test cases that are released in previous releases.

Figure 4-12 BSF_Regression Features Documentation



4.1.7 Running BSF_HealthCheck Pipeline

This is a pre-configured pipeline where ATS performs a test probe with SUT. It triggers helm test and provides the results in Jenkins Console Logs.

You can run BSF_HealthCheck pipeline to check if all BSF pods are up and running. If yes, it provides the status as successful. If any pod is down due to any reason, then the pipeline fails.

To configure the parameters for **BSF_HealthCheck** pipeline, perform the following steps:

1. Click **BSF_HealthCheck** in the **Name** column.
2. Click **Configure** in the left navigation pane.
3. When you scroll-down, the **General** tab becomes active. Be sure that the screen loads completely.
4. Continue scrolling down until the **Pipeline** tab becomes active. The following is a screen capture that shows the Pipeline script:

Figure 4-13 Helm Test Script

5. In the `Script` area under the Pipeline section, users may customize the values for the following parameters:

> **NOT_SUPPORTED**
>
> Do not modify values of parameters other than the ones described in this section.

   - **a:** Change this parameter to update the helm release name where BSF is deployed in your bastion.
   - **b:** Change this parameter to update the namespace where BSF is deployed in your bastion.

6. Click **Save** to update the values.

Running Helm Test

To run BSF test cases, click **Build Now**.

4.2 Running NSSF Test Cases using ATS

This section describes how to run NSSF test cases using ATS.

4.2.1 Prerequisites

The prerequisites to run NSSF Test Cases using NSSF ATS 25.2.200 are:

- Deploy NSSF 25.2.200 with default helm configurations using helm charts.
- All NSSF microservices must be up and running.
- Both NSSF ATS and ATS Deployment needs to be same namespace.
- For NSSF ATS 25.2.200, deploy one stub server and the service name should be **"amf-stubserver"**. It is required to run AMF-subscription Notification functionality test cases.
- For NSSF ATS 25.2.200, deploy one stub server and the service name should be **"nrf-stubserver"**. It is required to run NRF-subscription Notification functionality test cases.
- For NSSF ATS 25.2.200, deploy one stub server and the service name should be **"nrf-stubserver1"**. It is required to run NRF-selection based on DNS SRV.
- For NSSF ATS 25.2.200, deploy one stub server and the service name should be **"nrf-stubserver2"**. It is required to run NRF-selection based on DNS SRV.
- For NSSF 25.2.200, deploy one DNS stub server and the service name should be **"ocdns-bind"**. It is required to run NRF-selection based on DNS SRV.

4.2.2 Logging into ATS

Before logging into ATS, deploy ATS using HELM charts as shown below:

Verify ATS deployment

```
[opc@ocnssf-oci-phx-einstein-bastion-01 ~]$ helm status ocats

NAME: ocats
LAST DEPLOYED: Thu May 30 10:21:40 2024
NAMESPACE: cicdnssf-240530102024
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

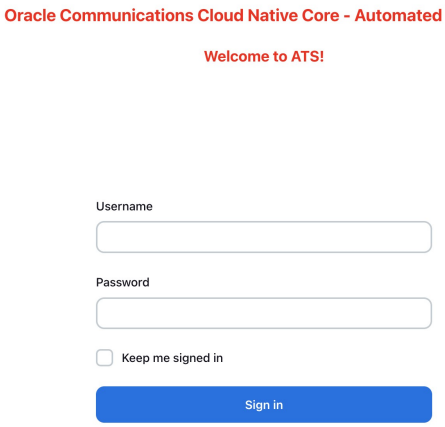
There are two ways to log in to ATS GUI.

- When an external load balancer (metalLB in case of OCCNE) is available and you provide an external IP to the ATS service, the user can log in to ATS GUI using <External-IP>:8080.
- When you do not provide an external IP to the ATS service, open the browser and enter the external IP of the worker node and nodeport of the ATS service to log in to the ATS GUI.
  `<Worker-Node-IP>:<Node-Port-of-ATS>`

> **Note**
>
> In the **Verifying ATS Deployment** screenshot, the ATS nodeport is highlighted in red as 32013. For more details on ATS deployment, refer to **NSSF ATS Installation Procedure.**

Open a browser and enter the IP Address and port details as <Worker-Node-IP>:<NodePort-of-ATS> (In the above example, the Worker-Node-IP and NodePort-of-ATS are 10.98.101.177:32013, which are shown as highlighted in the screenshot above).

The ATS login screen appears.

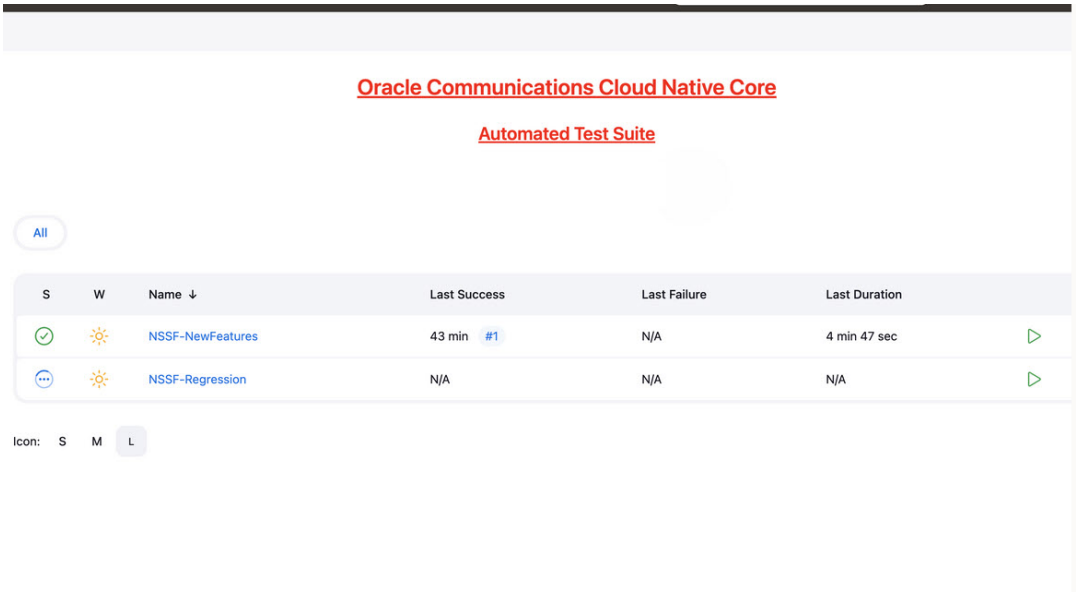


To run ATS:

1. Enter the username as 'nssfuser' and password as 'nssfpasswd'. Click **Sign in**. A page with preconfigured pipelines appears.

> **Note**
>
> To modify the default login password, refer to Modifying Login Password.

A page with preconfigured pipelines appears.

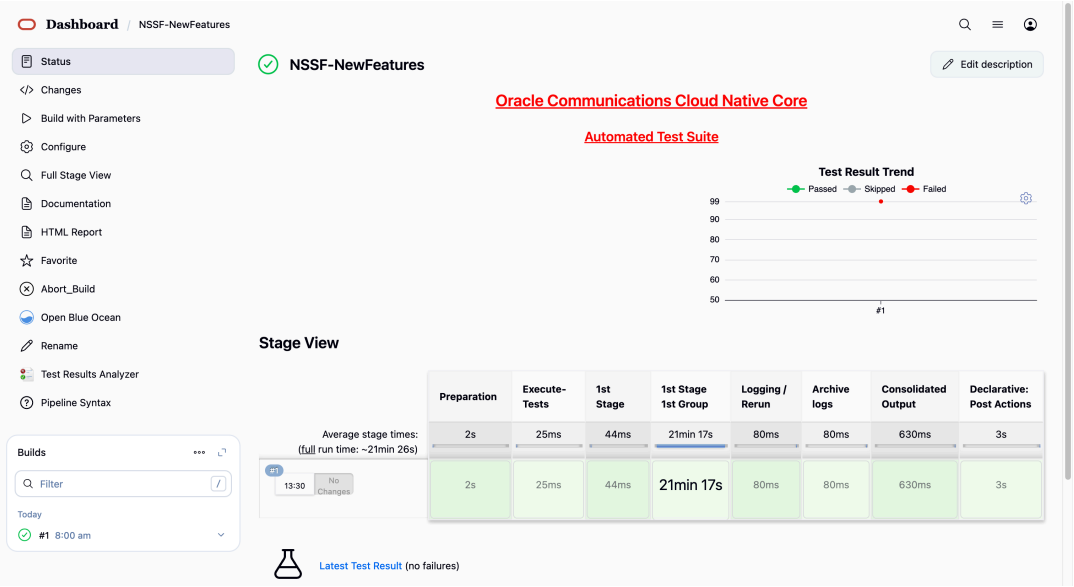


- **NSSF-New-Features:** This pipeline has all the test cases delivered as part of NSSF ATS - 25.2.200.
- **NSSF-Regression:** This pipeline has the test cases of all the previous releases.

4.2.3 NSSF-NewFeatures Pipeline

In this pipeline, you can configure ATS, which is a one-time activity as per System Under Test (SUT) deployment. You can also run all the new NSSF test cases using the pipeline. To configure its parameters:

1. Click **NSSF-NewFeatures** in the **Name** column. The following screen appears:



In the above screen:

   a. Click **Configure** to access the configuration screen.

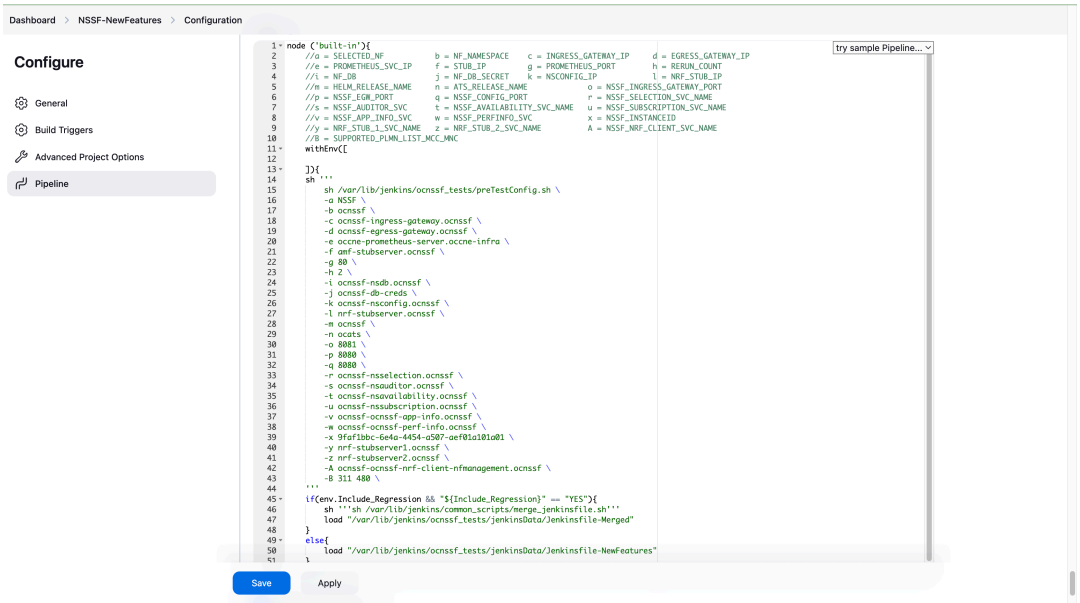
   b. Click **Documentation** to view the documented test cases.

   c. Click blue dots inside the **Build History** box to view the success console logs of the "All" and "Sanity" respectively.

   d. The **Stage View** represents the already run pipeline for customer reference.

2. Click **Configure**. Users MUST wait for the page to load completely. Once the page loads completely, click the **Pipeline** tab to reach the Pipeline configuration as shown below:

> **Warning**
>
> Make sure that the screen shown above loads completely before you perform any action on it. Also, do not modify any configuration other than that discussed below.



3. You can modify script pipeline parameters from "a" to "B" on the basis of your deployment environment and click Save. The content of the pipeline script is as follows:

```
node ('built-in'){
    //a = SELECTED_NF            b = NF_NAMESPACE    c =
INGRESS_GATEWAY_IP     d = EGRESS_GATEWAY_IP
    //e = PROMETHEUS_SVC_IP      f = STUB_IP         g =
PROMETHEUS_PORT        h = RERUN_COUNT
    //i = NF_DB                  j = NF_DB_SECRET    k =
NSCONFIG_IP            l = NRF_STUB_IP
    //m = HELM_RELEASE_NAME      n = ATS_RELEASE_NAME             o =
NSSF_INGRESS_GATEWAY_PORT
    //p = NSSF_EGW_PORT          q = NSSF_CONFIG_PORT             r =
NSSF_SELECTION_SVC_NAME
    //s = NSSF_AUDITOR_SVC       t = NSSF_AVAILABILITY_SVC_NAME   u =
NSSF_SUBSCRIPTION_SVC_NAME
    //v = NSSF_APP_INFO_SVC      w = NSSF_PERFINFO_SVC            x =
NSSF_INSTANCEID
    //y = NRF_STUB_1_SVC_NAME    z = NRF_STUB_2_SVC_NAME          A =
NSSF_NRF_CLIENT_SVC_NAME
    //B = SUPPORTED_PLMN_LIST_MCC_MNC
    withEnv([

    ]){
    sh '''
        sh /var/lib/jenkins/ocnssf_tests/preTestConfig.sh \
        -a NSSF \
        -b ocnssf \
        -c ocnssf-ingress-gateway.ocnssf \
        -d ocnssf-egress-gateway.ocnssf \
        -e occne-prometheus-server.occne-infra \
        -f amf-stubserver.ocnssf \
        -g 80 \
        -h 2 \
        -i ocnssf-nsdb.ocnssf \
```

```
            -j ocnssf-db-creds \
            -k ocnssf-nsconfig.ocnssf \
            -l nrf-stubserver.ocnssf \
            -m ocnssf \
            -n ocats \
            -o 8081 \
            -p 8080 \
            -q 8080 \
            -r ocnssf-nsselection.ocnssf \
            -s ocnssf-nsauditor.ocnssf \
            -t ocnssf-nsavailability.ocnssf \
            -u ocnssf-nssubscription.ocnssf \
            -v ocnssf-ocnssf-app-info.ocnssf \
            -w ocnssf-ocnssf-perf-info.ocnssf \
            -x 9faf1bbc-6e4a-4454-a507-aef01a101a01 \
            -y nrf-stubserver1.ocnssf \
            -z nrf-stubserver2.ocnssf \
            -A ocnssf-ocnssf-nrf-client-nfmanagement.ocnssf \
            -B 311 480 \
        '''
        if(env.Include_Regression && "${Include_Regression}" == "YES"){
            sh '''sh /var/lib/jenkins/common_scripts/merge_jenkinsfile.sh'''
            load "/var/lib/jenkins/ocnssf_tests/jenkinsData/Jenkinsfile-Merged"
        }
        else{
            load "/var/lib/jenkins/ocnssf_tests/jenkinsData/Jenkinsfile-
NewFeatures"
        }
        }
}
```

> **Note**
>
> The User MUST NOT change any other value apart from these parameters.

The description of these parameters is as follows:

- a: Name of the NF to be tested in capital (NSSF).
- b: Namespace in which the NSSF is deployed (default is ocnssf)
- c: Ingress Gateway IP address (default is ocnssf-ingress-gateway.ocnssf)
- d: Egress Gateway IP address (default is ocnssf-egress-gateway.ocnssf)
- e: Prometheus service IP address (default is prometheus.cne-infra)
- f: Stub service IP address (default is ocats-amf-stubserver.ocnssf)
- g: Port of Prometheus service (default is 80)
- h: Number of times the re-run of failed case is allowed (default is 2).
- i: Database name (default is ocnssf-nsdb.ocnssf)
- j Database secrets (default ocnssf-db-creds)
- k: NSSF config ip address (ocnssf-nsconfig.ocnssf)
- l: NRF stub server IP address (ocats-nrf-stubserver.ocnssf)

- m: NSSF release name (ocnssf)
- n: ATS release name (ocats)
- o: NSSF Ingress Gateway Port
- p: NSSF Egress Gateway Port
- q: NSSF Config Port
- r: NSSF Selection Service Name
- s: NSSF Auditor Service Name
- t: NSSF Availability Service Name
- u: NSSF Subscription Service Name
- v: APP Info Service Name
- w:Perf info Service Name
- x: NSSF Supported PLMN
- A: NRF Client Managment Service Name (ocnssf-ocnssf-nrf-client-nfmanagement.ocnssf)
- B: NSSF PLMN List (311 480)

Note

- Do not change any value if the OCCNE cluster is used and NSSF, ATS, and STUB are deployed in the ocnssf namespace.
- In the above image NSSF Helm release name is "ocnssf", ATS Helm Release Name is "ocats" and namespace is "ocnssf". If any change in helm release name of NSSF, ATS and namespace needs to be updated accordingly.
  For example, NSSF Helm release name is "ocnssfats" and ATS Helm release name is "ocatsnssf" and namespace is "ocnssf2510", then above Pipeline Configuration should be edited as shown below.
- NSSF Helm release name should be updated in "c d i k m r s t u v w" if any change in NSSF helm release name apart from "ocnssf".
- The NSSF Instance ID needs to be updated in the ATS Jenkins pipeline parameters if there are any changes in the NSSF custom values (CV) file. By default, the NSSF Instance ID is set to "`9faf1bbc-6e4a-4454-a507-aef01a101a01`". If the Instance ID is modified in the NSSF CV file, the same changes must be reflected in the Jenkins pipeline parameters.
  By default, NSSF Instance ID in NSSF custom values file as below:

  ```
  #InstanceId of NSSF used in case of GR
  nfInstanceId: &nfInstanceId "9faf1bbc-6e4a-4454-a507-
  aef01a101a01"
  ```

  For example, if the nfInstanceId in the NSSF custom values file is modified from "`9faf1bbc-6e4a-4454-a507-aef01a101a01`" to "`9faf1bbc-6e4a-4454-a507-aef01a101a20`", the same change must be updated in the "x" Jenkins pipeline parameters as shown below:

  ```
  node ('built-in'){
      //a = SELECTED_NF           b = NF_NAMESPACE    c =
  INGRESS_GATEWAY_IP     d = EGRESS_GATEWAY_IP
      //e = PROMETHEUS_SVC_IP     f = STUB_IP         g =
  PROMETHEUS_PORT        h = RERUN_COUNT
      //i = NF_DB                 j = NF_DB_SECRET    k =
  NSCONFIG_IP            l = NRF_STUB_IP
      //m = HELM_RELEASE_NAME     n = ATS_RELEASE_NAME
  o = NSSF_INGRESS_GATEWAY_PORT
      //p = NSSF_EGW_PORT         q = NSSF_CONFIG_PORT
  r = NSSF_SELECTION_SVC_NAME
      //s = NSSF_AUDITOR_SVC      t = NSSF_AVAILABILITY_SVC_NAME
  u = NSSF_SUBSCRIPTION_SVC_NAME
      //v = NSSF_APP_INFO_SVC     w = NSSF_PERFINFO_SVC
  x = NSSF_INSTANCEID
      //y = NRF_STUB_1_SVC_NAME   z = NRF_STUB_2_SVC_NAME
  A = NSSF_NRF_CLIENT_SVC_NAME
      //B = SUPPORTED_PLMN_LIST_MCC_MNC
      withEnv([

      ]){
      sh '''
          sh /var/lib/jenkins/ocnssf_tests/preTestConfig.sh \
          -a NSSF \
  ```

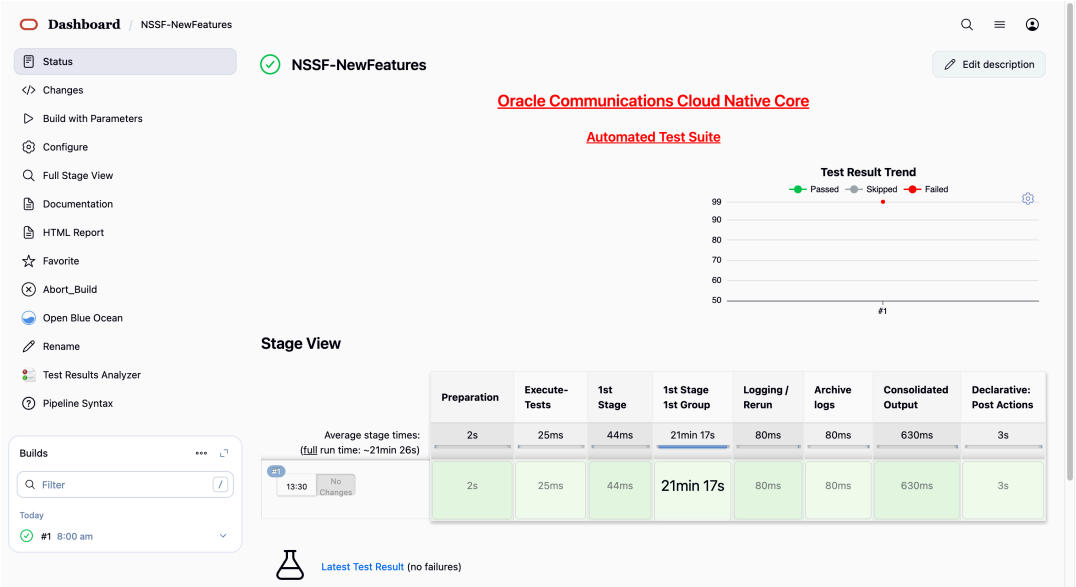
```
                -b ocnssf2510 \
                -c ocnssfats-ingress-gateway.ocnssf2510 \
                -d ocnssfats-egress-gateway.ocnssf2510 \
                -e occne-prometheus-server.occne-infra \
                -f amf-stubserver.ocnssf2510 \
                -g 80 \
                -h 2 \
                -i ocnssf-nsdb.ocnssf \
                -j ocnssf-db-creds \
                -k ocnssfats-nsconfig.ocnssf2510 \
                -l nrf-stubserver.ocnssf2510 \
                -m ocnssfats \
                -n ocats \
                -o 8081 \
                -p 8080 \
                -q 8080 \
                -r ocnssfats-nsselection.ocnssf2510 \
                -s ocnssfats-nsauditor.ocnssf2510 \
                -t ocnssfats-nsavailability.ocnssf2510 \
                -u ocnssfats-nssubscription.ocnssf2510 \
                -v ocnssfats-ocnssf-app-info.ocnssf2510 \
                -w ocnssfats-ocnssf-perf-info.ocnssf2510 \
                -x 9faf1bbc-6e4a-4454-a507-aef01a101a20 \
                -y nrf-stubserver1.ocnssf2510 \
                -z nrf-stubserver2.ocnssf2510 \
                -A ocnssfats-ocnssf-nrf-client-nfmanagement.ocnssf2510 \
                -B 311 480 \
            '''
            if(env.Include_Regression && "${Include_Regression}" ==
        "YES"){
                sh '''sh /var/lib/jenkins/common_scripts/
        merge_jenkinsfile.sh'''
                load "/var/lib/jenkins/ocnssf_tests/jenkinsData/
        Jenkinsfile-Merged"
            }
            else{
                load "/var/lib/jenkins/ocnssf_tests/jenkinsData/
        Jenkinsfile-NewFeatures"
            }
            }
        }
```

4. Click **Save** after making necessary changes. The **Pipeline NSSF-NewFeatures** screen appears.

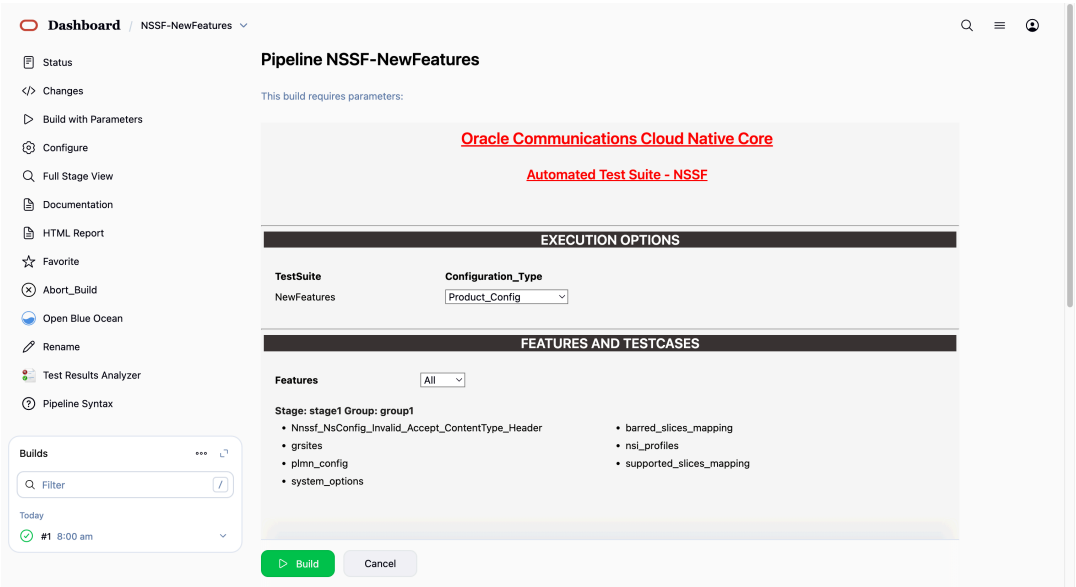
Running NSSF New Features Test Cases

To run NSSF New Features test cases:

1. Go back to new features and Click **Build with Parameters** present on the NSSF-NewFeatures screen in the extreme left column corresponding to NSSF-NewFeatures row as shown below:

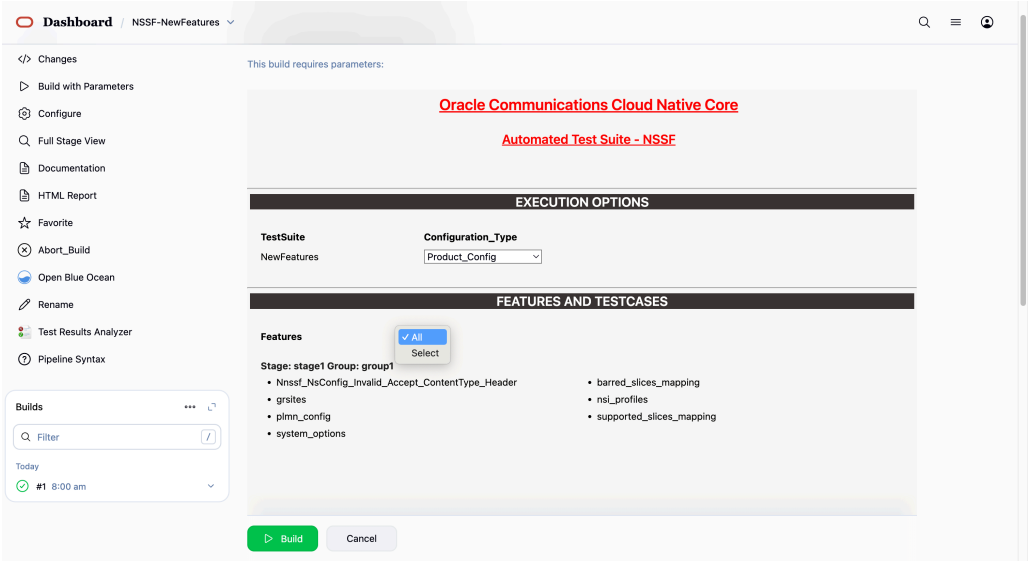


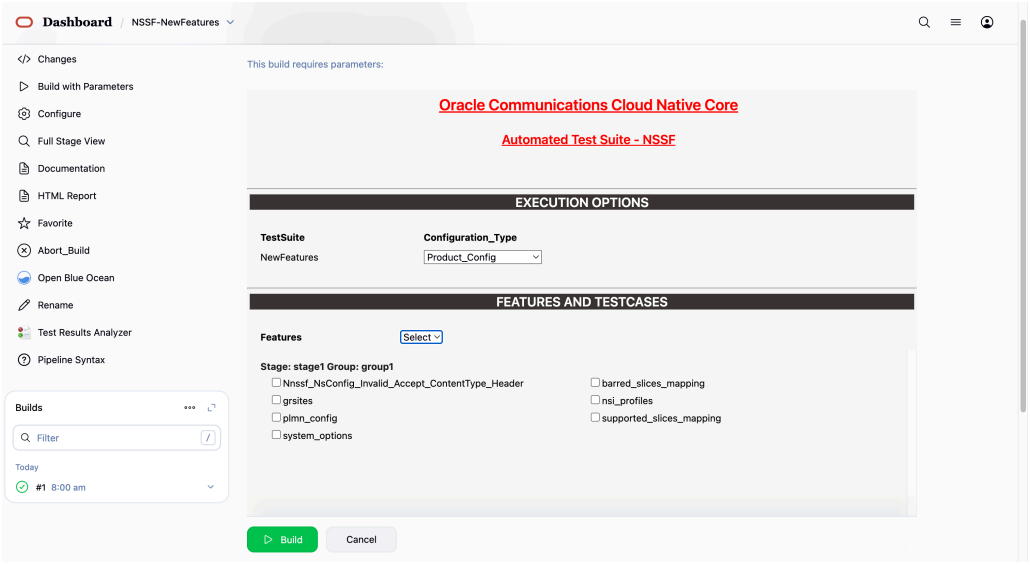
2. The following screen appears:



In the above screen, there are three **Select_Option(s)**, which are:

- By default, features 'ALL', configuration type 'Product_config', include regression 'NO' will be selected, and all test cases will be executed once clicked on BUILD, as shown above.
- Features: ALL. Once you click on ALL, a dropdown 'Select' option will appear. After selecting 'Select,' options will appear to choose feature files. To run the selected feature files, click on 'Build,' as shown below.



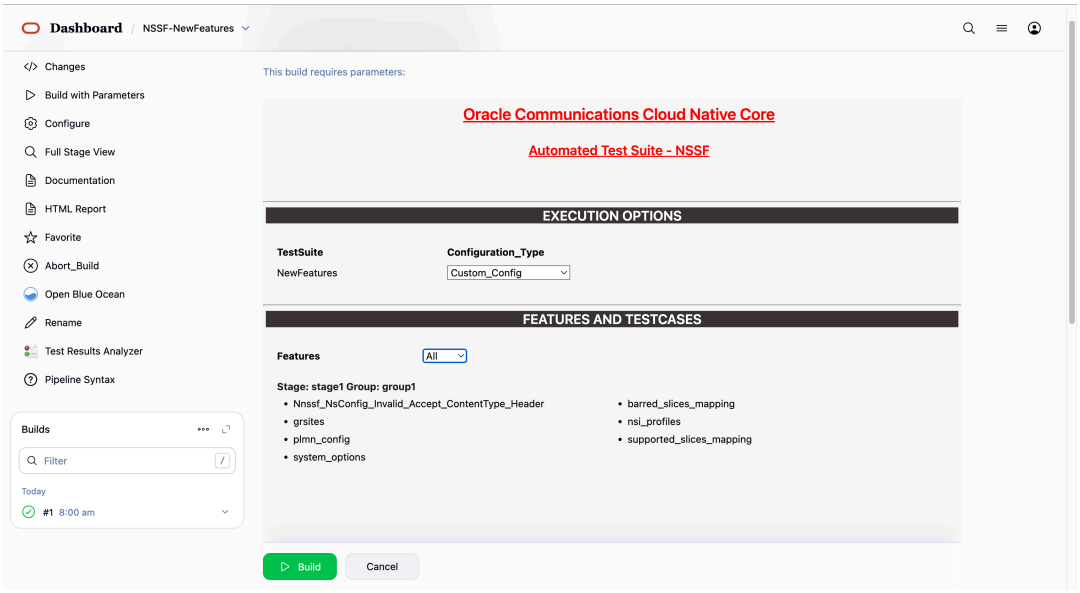


3. Select one of the following configuration types:
   - **Product_Config**: On selecting this option, test cases from product folders are populated on ATS UI and product configuration is applied to them via the keyvalue pair and yaml files defined or present in the "Product Config" folder.
   - **Custom_Config**: On selecting this option, test cases from custom folders are populated on ATS UI and custom configuration is applied to them via the keyvalue pair and yaml files defined or present in the "Custom Config" folder. To use the Parameterization feature, always select the Custom_Config option. User can copy, add, or delete the required test cases that are available for the NSSF and place them appropriately within the custom folder for NSSF-NewFeatures. Reload the page to view the test cases available in the custom NewFeatures folder. For more information, see Parameterized approach for SUT custom configuration.

   The NSSF test cases are divided into NSSF Service operations as follows:
   - NSSF_204NoContent_NsAvailability_PATCH_REMOVE- This feature file contains test cases related to PATCH or PUT request for deleting all slices.

- Virtual_Host_NRF_Resolution_By_NSSF_Using_DNSSRV- This feature file contains test cases related to DNS SRV based selection of NRF in NSSF.

If you want to run custom_config, select below parameter as custom_config, where the framework automatically points out to custom_config and cust_data.




```
7 features passed, 0 failed, 0 skipped
99 scenarios passed, 0 failed, 0 skipped
3079 steps passed, 0 failed, 0 skipped, 0 undefined
Took 21m10.088s
```

Parameterized approach for SUT custom configuration

Using this feature, user can make the following customizations to custom folders:

- Add new test cases by adding datafiles
- Remove test cases
- Modify the parameters and their values in the key-value pair or `<feature>`.yaml files

To run ATS test cases, user can maintain as many versions of Custom_Config folder by using the following naming convention:

Cust ConfigN

where N can be any number

At the time of execution, ensure to rename the required folder to Custom Config folder as Jenkins always retrieves data from this folder when user selects Custom_Config.

Updating Global Parameters

To use Custom_Config, it is required to change the value of cust_folder from data to cust_data in global.yaml file. In addition, you can customize the parameters and their respective values in the global.yaml as per the requirements.

Updating Feature Parameters

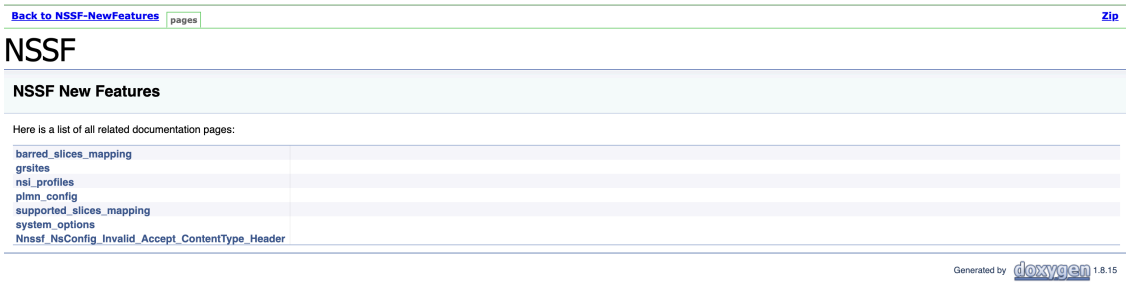
Consider the following points when customizing <feature>.yaml files for parameterized feature:

- In addition to global.yaml parameters, feature files may also contain parameters for which user can update values at the time of running pipelines.
- Changing the values of parameters tagged as "Feature Specific Value" may cause failures at the time of running pipelines.
- Values for parameters tagged with #START_GLOBAL and #END_GLOBAL tags take values from global.yaml.

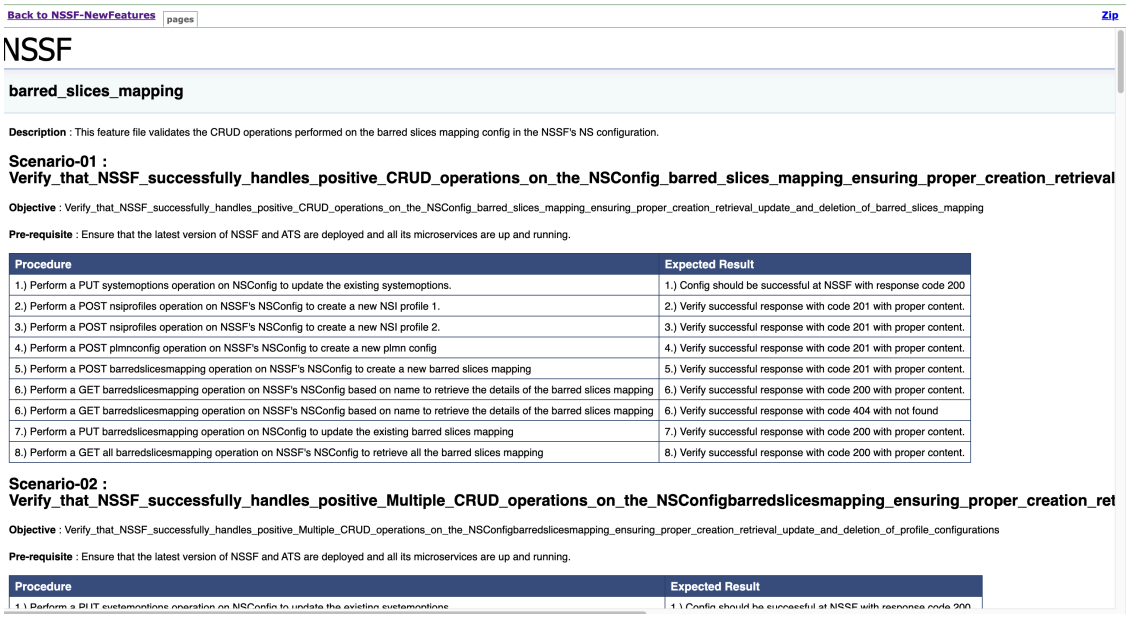
> **Note**
>
> For NSSF-ATS release 25.2.200, parameterization is supported for NSSF NewFeatures pipeline only.

4.2.4 NSSF-NewFeatures Documentation

To view NSSF functionalities, go to **NSSF-NewFeatures** pipeline latest build and click the **Documentation** link in the left navigation pane. The following screen appears. Click any functionality to view its test cases and scenarios of each test case as shown in the sample screenshot below:



A sample of a few documentation features are as follows:



Back to NSSF-NewFeatures pages Zip

NSSF

barred_slices_mapping

Description : This feature file validates the CRUD operations performed on the barred slices mapping config in the NSSF's NS configuration.

Scenario-01 :
Verify_that_NSSF_successfully_handles_positive_CRUD_operations_on_the_NSConfig_barred_slices_mapping_ensuring_proper_creation_retrieval

Objective : Verify_that_NSSF_successfully_handles_positive_CRUD_operations_on_the_NSConfig_barred_slices_mapping_ensuring_proper_creation_retrieval_update_and_deletion_of_barred_slices_mapping

Pre-requisite : Ensure that the latest version of NSSF and ATS are deployed and all its microservices are up and running.

Procedure	Expected Result
1.) Perform a PUT systemoptions operation on NSConfig to update the existing systemoptions.	1.) Config should be successful at NSSF with response code 200
2.) Perform a POST nsiprofiles operation on NSSF's NSConfig to create a new NSI profile 1.	2.) Verify successful response with code 201 with proper content.
3.) Perform a POST nsiprofiles operation on NSSF's NSConfig to create a new NSI profile 2.	3.) Verify successful response with code 201 with proper content.
4.) Perform a POST plmnconfig operation on NSSF's NSConfig to create a new plmn config	4.) Verify successful response with code 201 with proper content.
5.) Perform a POST barredslicesmapping operation on NSSF's NSConfig to create a new barred slices mapping	5.) Verify successful response with code 201 with proper content.
6.) Perform a GET barredslicesmapping operation on NSSF's NSConfig based on name to retrieve the details of the barred slices mapping	6.) Verify successful response with code 200 with proper content.
6.) Perform a GET barredslicesmapping operation on NSSF's NSConfig based on name to retrieve the details of the barred slices mapping	6.) Verify successful response with code 404 with not found
7.) Perform a PUT barredslicesmapping operation on NSConfig to update the existing barred slices mapping	7.) Verify successful response with code 200 with proper content.
8.) Perform a GET all barredslicesmapping operation on NSSF's NSConfig to retrieve all the barred slices mapping	8.) Verify successful response with code 200 with proper content.

Scenario-02 :
Verify_that_NSSF_successfully_handles_positive_Multiple_CRUD_operations_on_the_NSConfigbarredslicesmapping_ensuring_proper_creation_ret

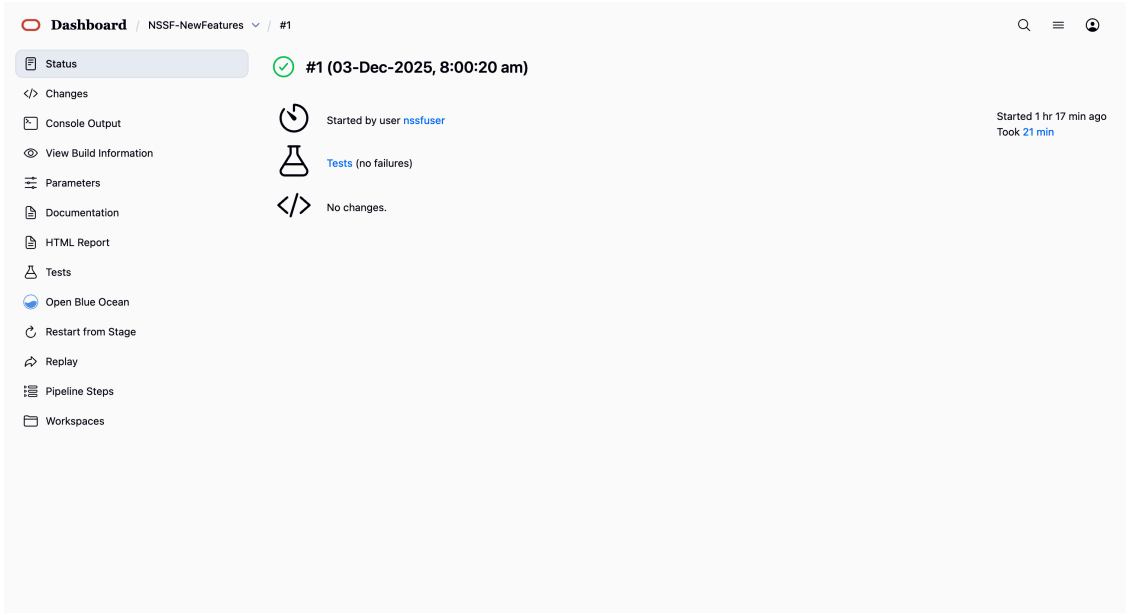
Objective : Verify_that_NSSF_successfully_handles_positive_Multiple_CRUD_operations_on_the_NSConfigbarredslicesmapping_ensuring_proper_creation_retrieval_update_and_deletion_of_profile_configurations

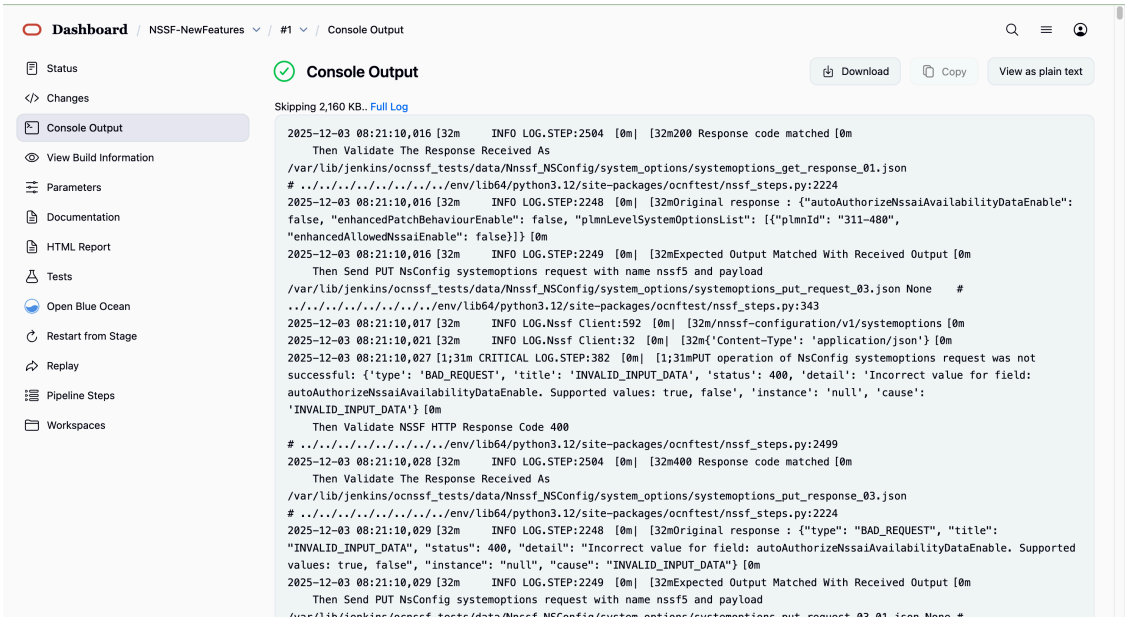
Pre-requisite : Ensure that the latest version of NSSF and ATS are deployed and all its microservices are up and running.

Procedure	Expected Result
1.) Perform a PUT systemoptions operation on NSConfig to update the existing systemoptions.	1.) Config should be successful at NSSF with response code 200

Once the run of new features done successfully, click on the build number whose logs are to be viewed and then click on **Console Output** on left side navigation.

Wait till the page is loaded completely, then click **Download** to download the new features logs.





4.2.5 NSSF-Regression Pipeline

This pipeline contains test cases from the previous versions.

Some of the test cases are updated as per the new implementation of NSSF.

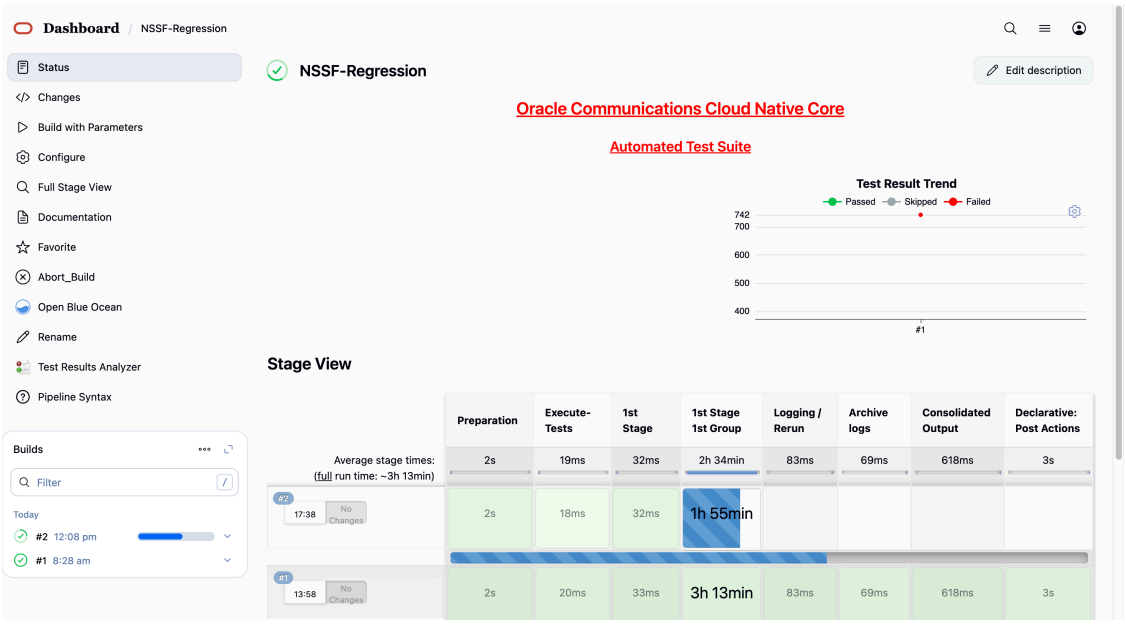
The configuration method and parameters are the same as the **NewFeatures** pipeline.

The NSSF test cases are divided into NSSF Service operations. The classification is as follows:

- **NSSF_Sending_Notification_via_ARS_SCP:** Tests NSSF's ability to route and send notifications to SCP(s) via ARS, including single/multiple SCPs, priorities, virtual hosts, alternate routes, and related error handling.
- **NSSF_User_Agent_Header:** Validates the User-Agent headers based on whether the feature is enabled or disabled.
- **oauth_signValidationServiceMeshEnabled_false:** Validates CRUD operations performed on OAuth in NSSF's Ingress Gateway configuration.
- **NSSF_OAuth_Response_Header:** Ensures NSSF sends correct OAuth2-related HTTP error codes and headers for invalid, expired, or insufficient-scope OAuth2 tokens.
- **NSSF_Server_Header:** Validates the Server Header feature.
- **NSSF_SCP_Monitoring_OPTIONS:** Tests NSSF SCP peer monitoring and routing logic, including configuration, healthPath enforcement, priorities, and error handling for SBI and indirect routing scenarios.
- **Nnssf_NSSelection_PDU_Miscellaneous_AutoConfiguration_ON:** Tests NSSelection with auto-configuration ON for PDU session flows, configuration changes, and error scenarios.
- **Nnssf_NSSelection_UE_CU_Invalid_Query_Parameters:** Tests invalid query parameter scenarios for UE/CU NSSelection requests.
- **Nnssf_NSSelection_Initial_Reg_AutoConfiguration_ON_EnhancedNssai_ON:** Validates NSSelection behavior when both autoAuthorizeNssaiAvailabilityDataEnable and enhancedAllowedNssaiEnable are True.

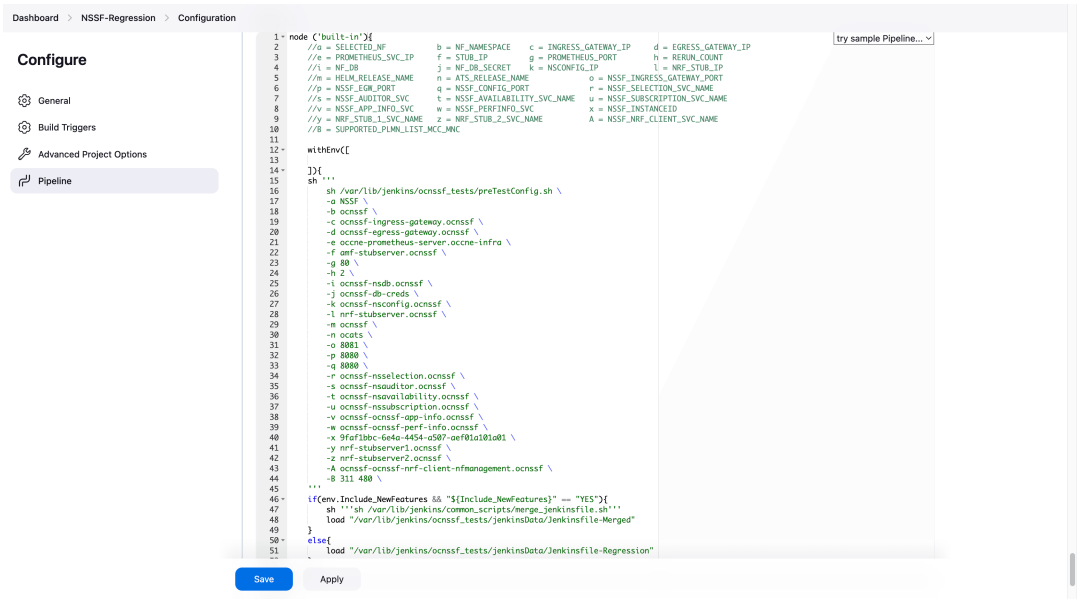
- **Nnssf_NSSelection_PDU_AutoConfiguration_ON:** Validates NSSelection PDU auto-configuration ON scenarios.
- **Nnssf_NSSelection_PDU_AutoConfiguration_OFF:** Validates NSSelection PDU auto-configuration OFF scenarios.
- **Nnssf_NSSelection_PDU_Miscellaneous_AutoConfiguration_OFF:** Tests NSSelection PDU miscellaneous cases with auto-configuration OFF, including PDU session establishment and error handling.
- **Nnssf_NSSelection_UE_CU_AutoConfiguration_OFF_EnhancedNssai_OFF:** Validates NSSelection when both autoAuthorizeNssaiAvailabilityDataEnable and enhancedAllowedNssaiEnable are False.
- **Nnssf_NSSelection_Invalid_Accept_ContentType_Header:** Tests rejection of NSSelection GET requests with invalid or missing Accept/Content-Type headers.
- **Nnssf_NSSelection_Initial_Reg_AutoConfiguration_OFF_EnhancedNssai_OFF:** Validates NSSelection for initial registration with auto-configuration OFF and enhancedNssai OFF.
- **Nnssf_NSSelection_UE_CU_AutoConfiguration_ON_EnhancedNssai_OFF:** Validates NSSelection when auto-configuration is ON and enhancedNssai is OFF.
- **Nnssf_NSSelection_PDU_Session_Invalid_Query_Parameters:** Validates invalid query parameters for PDU session NSSelection.
- **NSSF_Unknown_URI_Unsupported_Method:** Validates CRUD behavior for unknown URIs and unsupported HTTP methods.
- **Nnssf_NSSelection_Initial_Reg_AutoConfiguration_OFF_EnhancedNssai_ON:** Validates initial registration NSSelection with auto-configuration OFF and enhancedNssai ON.
- **Nnssf_NSSelection_Initial_Reg_Invalid_Query_Parameters:** Validates invalid query parameter handling for initial registration NSSelection.
- **Nnssf_NSSelection_UE_CU_AutoConfiguration_OFF_EnhancedNssai_ON:** Validates NSSelection when auto-configuration is OFF and enhancedNssai is ON.
- **Nnssf_NSSelection_UE_CU_AutoConfiguration_ON_EnhancedNssai_ON:** Validates NSSelection when both auto-configuration and enhancedNssai are ON.
- **Nnssf_NSSelection_Initial_Reg_AutoConfiguration_ON_EnhancedNssai_OFF:** Validates initial registration NSSelection with auto-configuration ON and enhancedNssai OFF.
- **NSSF_NSAvailability_OPTIONS:** Tests NSSF responses to OPTIONS requests for supported communication methods.
- **Nnssf_Nssai_Availability_patch_with_autoconfiguration:** Validates PATCH operations on NS availability with auto-configuration enabled.
- **Nnssf_Nssai_Availability_put_patch_with_supported_features:** Validates PUT and PATCH operations in NS availability with supported features.
- **Nnssf_Nssai_Availability_put_delete_with_autoconfiguration:** Validates PUT and DELETE operations on NS availability with auto-configuration enabled.
- **Nnssf_NSAvailability_Invalid_Accept_ContentType_Header:** Validates rejection of NSAvailability requests with invalid or missing Accept/Content-Type headers.
- **Nnssf_Nssai_Availability_patch_remove_204_NoContent:** Validates PATCH operations with and without auto-configuration and enhanced patch behavior, ensuring correct 204 responses.

- **Nnssf_Nssai_Availability_put_delete_without_autoconfiguration:** Validates PUT and DELETE NS availability operations with auto-configuration disabled.
- **Nnssf_Nssai_Availability_patch_without_autoconfiguration:** Validates PATCH operations on NS availability without auto-configuration.
- **Nnssf_Nssai_Availability_put_patch_delete_error_scenarios:** Validates error scenarios for PUT, PATCH, and DELETE operations in NSAvailability with auto-configuration enabled/disabled.
- **NSSF_Sanity:** Validates NSSF sanity test cases.
- **Nnssf_NSSubscription_Notification_Response_Handling:** Validates NSSF subscription notification behavior based on various AMF stub responses.
- **Nnssf_NSSubscription_Notification_Auto_Config_OFF:** Validates notification behavior when auto-configuration is OFF.
- **Nnssf_NSSubscription_Invalid_Accept_ContentType_Header:** Validates subscription request handling with invalid Accept/Content-Type headers.
- **NSSF_Subscription_Delete:** Validates deletion operations and error handling for NSSubscription resources.
- **Nnssf_NSSubscription_Indirect_Communication_Negative:** Validates negative scenarios for indirect communication with incorrect 3gpp-Sbi-Binding headers.
- **Nnssf_NSSubscription_Subscription_Invalid_Params:** Validates CRUD operations and parameter validation for NSSubscription requests.
- **Nnssf_NSSubscription_Indirect_Communication:** Validates indirect communication for the NSSubscription service.
- **Nnssf_NSSubscription_Subscription_Update:** Validates CRUD operations and input validations for NSSubscription updates.
- **Nnssf_NSSubscription_Notification_Auto_Config_ON:** Validates notification behavior when auto-configuration is ON.
- **Nnssf_NSSubscription_Post:** Validates CRUD operations and request handling for NSSubscription POST requests.
- **Virtual_Host_NRF_Resolution_By_NSSF_Using_DNSSRV:** Validates virtual host NRF resolution by NSSF using DNS SRV.
- **Multiple_PLMN**: Validates multiple PLMN:



1. In the above screen:
   - Click **Configure** to access the configuration screen.
   - Click **Documentation** to view the documented test cases.
   - Click blue dots inside the **Build History** box to view the success console logs of the "All" and "Sanity" respectively.
   - The **Stage View** represents the already run pipeline for customer reference.
2. Click **Configure**. Users MUST wait for the page to load completely. Once the page loads completely, click the **Pipeline** tab to reach the Pipeline configuration as shown below:

> **Warning**
>
> Make sure that the screen shown above loads completely before you perform any action on it. Also, do not modify any configuration other than that discussed below.



3. You can modify script pipeline parameters from "a" to "B" on the basis of your deployment environment and click Save. The content of the pipeline script is as follows:

```
node ('built-in'){
    //a = SELECTED_NF              b = NF_NAMESPACE     c =
INGRESS_GATEWAY_IP     d = EGRESS_GATEWAY_IP
    //e = PROMETHEUS_SVC_IP      f = STUB_IP          g =
PROMETHEUS_PORT        h = RERUN_COUNT
    //i = NF_DB                  j = NF_DB_SECRET     k =
NSCONFIG_IP            l = NRF_STUB_IP
    //m = HELM_RELEASE_NAME      n = ATS_RELEASE_NAME             o =
NSSF_INGRESS_GATEWAY_PORT
    //p = NSSF_EGW_PORT          q = NSSF_CONFIG_PORT             r =
NSSF_SELECTION_SVC_NAME
    //s = NSSF_AUDITOR_SVC       t = NSSF_AVAILABILITY_SVC_NAME   u =
NSSF_SUBSCRIPTION_SVC_NAME
    //v = NSSF_APP_INFO_SVC      w = NSSF_PERFINFO_SVC            x =
NSSF_INSTANCEID
    //y = NRF_STUB_1_SVC_NAME    z = NRF_STUB_2_SVC_NAME          A =
NSSF_NRF_CLIENT_SVC_NAME
    //B = SUPPORTED_PLMN_LIST_MCC_MNC

    withEnv([

    ]){
    sh '''
        sh /var/lib/jenkins/ocnssf_tests/preTestConfig.sh \
        -a NSSF \
        -b ocnssf \
        -c ocnssf-ingress-gateway.ocnssf \
        -d ocnssf-egress-gateway.ocnssf \
        -e occne-prometheus-server.occne-infra \
        -f amf-stubserver.ocnssf \
        -g 80 \
        -h 2 \
```

```
            -i ocnssf-nsdb.ocnssf \
            -j ocnssf-db-creds \
            -k ocnssf-nsconfig.ocnssf \
            -l nrf-stubserver.ocnssf \
            -m ocnssf \
            -n ocats \
            -o 8081 \
            -p 8080 \
            -q 8080 \
            -r ocnssf-nsselection.ocnssf \
            -s ocnssf-nsauditor.ocnssf \
            -t ocnssf-nsavailability.ocnssf \
            -u ocnssf-nssubscription.ocnssf \
            -v ocnssf-ocnssf-app-info.ocnssf \
            -w ocnssf-ocnssf-perf-info.ocnssf \
            -x 9faf1bbc-6e4a-4454-a507-aef01a101a01 \
            -y nrf-stubserver1.ocnssf \
            -z nrf-stubserver2.ocnssf \
            -A ocnssf-ocnssf-nrf-client-nfmanagement.ocnssf \
            -B 311 480 \
        '''
        if(env.Include_NewFeatures && "${Include_NewFeatures}" == "YES"){
            sh '''sh /var/lib/jenkins/common_scripts/merge_jenkinsfile.sh'''
            load "/var/lib/jenkins/ocnssf_tests/jenkinsData/Jenkinsfile-Merged"
        }
        else{
            load "/var/lib/jenkins/ocnssf_tests/jenkinsData/Jenkinsfile-
Regression"
        }
        }
}
```

> **Note**
>
> The User MUST NOT change any other value apart from these parameters.

The description of these parameters is as follows:

- a: Name of the NF to be tested in capital (NSSF).
- b: Namespace in which the NSSF is deployed (default is ocnssf)
- c: Ingress Gateway IP address (default is ocnssf-ingress-gateway.ocnssf)
- d: Egress Gateway IP address (default is ocnssf-egress-gateway.ocnssf)
- e: Prometheus service IP address (default is prometheus.cne-infra)
- f: Stub service IP address (default is ocats-amf-stubserver.ocnssf)
- g: Port of Prometheus service (default is 80)
- h: Number of times the re-run of failed case is allowed (default is 2).
- i: Database name (default is ocnssf-nsdb.ocnssf)
- j Database secrets (default ocnssf-db-creds)
- k: NSSF config ip address (ocnssf-nsconfig.ocnssf)

- l: NRF stub server IP address (ocats-nrf-stubserver.ocnssf)
- m: NSSF release name (ocnssf)
- n: ATS release name (ocats)
- o: NSSF Ingress Gateway Port
- p: NSSF Egress Gateway Port
- q: NSSF Config Port
- r: NSSF Selection Service Name
- s: NSSF Auditor Service Name
- t: NSSF Availability Service Name
- u: NSSF Subscription Service Name
- v: APP Info Service Name
- w:Perf info Service Name
- x: NSSF NF Instance ID
- y: NRF stub Server Service name(nrf-stubserver1.ocnssf)
- z: NRF stub Server Service name(nrf-stubserver1.ocnssf)
- A: NRF Client Managment Service Name (ocnssf-ocnssf-nrf-client-nfmanagement.ocnssf)
- B: NSSF PLMN List (311 480)

Note

- Do not change any value if the OCCNE cluster is used, and NSSF, ATS, and STUB are deployed in the ocnssf namespace.
- In the above image, the NSSF Helm Release Name is "ocnssf", ATS Helm Release Name is "ocats", and Namespace is "ocnssf". If any change in the helm release name of NSSF, ATS, and namespace needs to be updated accordingly.
- For example, if the NSSF Helm Release Name is "ocnssfats" and ATS Helm Release Name is "ocatsnssf", and Namespace is "ocnssf2510", then the above Pipeline Configuration should be edited as shown below:
  - NSSF Helm Release Name should be updated in "c d i k m r s t u v w" if any change in NSSF helm release name apart from "ocnssf".
- The NSSF Instance ID needs to be updated in the ATS Jenkins pipeline parameters if there are any changes in the NSSF custom values (CV) file. By default, the NSSF Instance ID is set to "`9faf1bbc-6e4a-4454-a507-aef01a101a01`". If the Instance ID is modified in the NSSF CV file, the same changes must be reflected in the Jenkins pipeline parameters.
  By default NSSF Instance ID in NSSF custom values file as below:

  ```
  #InstanceId of NSSF used in case of GR
  nfInstanceId: &nfInstanceId "9faf1bbc-6e4a-4454-a507-
  aef01a101a01"
  ```

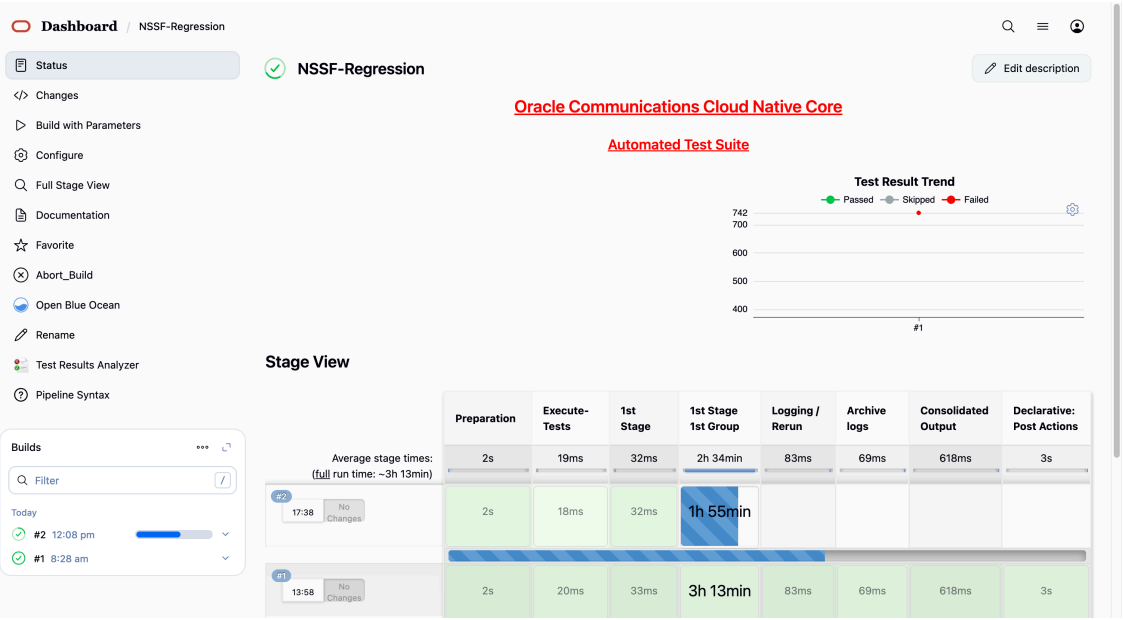
  For example, if the nfInstanceId in the NSSF custom values file is modified from "`9faf1bbc-6e4a-4454-a507-aef01a101a01`" to "`9faf1bbc-6e4a-4454-a507-aef01a101a20`", the same change must be updated in the "x" Jenkins pipeline parameters as shown below.

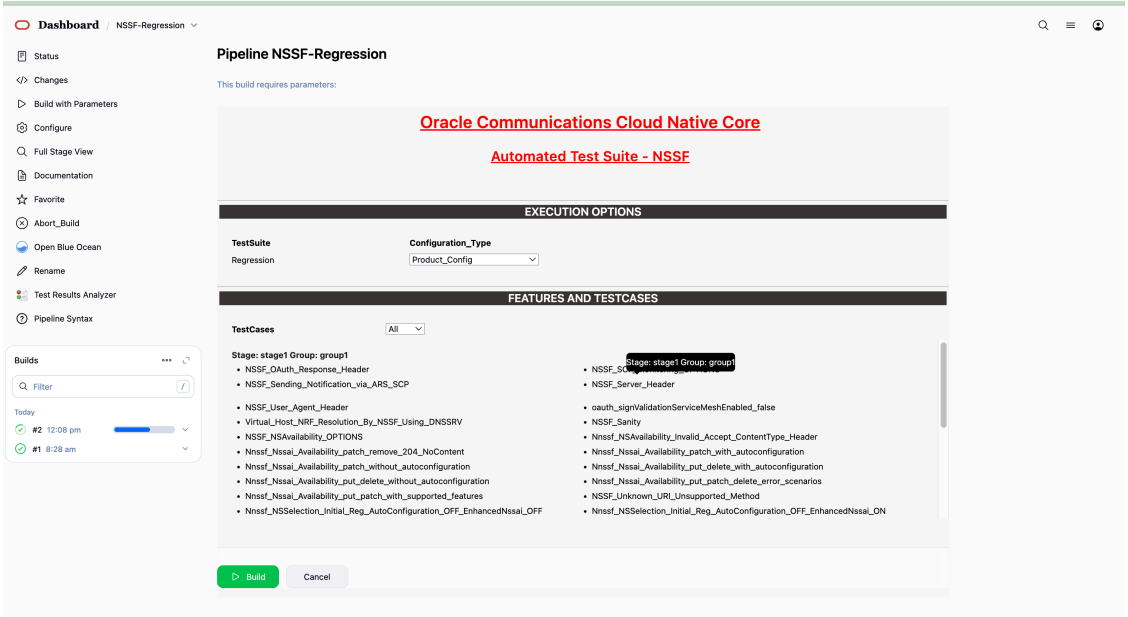
```
node ('built-in'){
    //a = SELECTED_NF           b = NF_NAMESPACE    c =
INGRESS_GATEWAY_IP     d = EGRESS_GATEWAY_IP
    //e = PROMETHEUS_SVC_IP     f = STUB_IP         g =
PROMETHEUS_PORT        h = RERUN_COUNT
    //i = NF_DB                 j = NF_DB_SECRET    k =
NSCONFIG_IP            l = NRF_STUB_IP
    //m = HELM_RELEASE_NAME     n = ATS_RELEASE_NAME             o
= NSSF_INGRESS_GATEWAY_PORT
    //p = NSSF_EGW_PORT         q = NSSF_CONFIG_PORT             r
= NSSF_SELECTION_SVC_NAME
    //s = NSSF_AUDITOR_SVC      t = NSSF_AVAILABILITY_SVC_NAME   u
= NSSF_SUBSCRIPTION_SVC_NAME
    //v = NSSF_APP_INFO_SVC     w = NSSF_PERFINFO_SVC            x
= NSSF_INSTANCEID
    //y = NRF_STUB_1_SVC_NAME   z = NRF_STUB_2_SVC_NAME          A
= NSSF_NRF_CLIENT_SVC_NAME
    //B = SUPPORTED_PLMN_LIST_MCC_MNC
    withEnv([

    ]){
    sh '''
        sh /var/lib/jenkins/ocnssf_tests/preTestConfig.sh \
        -a NSSF \
```

```
            -b ocnssf2510 \
            -c ocnssfats-ingress-gateway.ocnssf2510 \
            -d ocnssfats-egress-gateway.ocnssf2510 \
            -e occne-prometheus-server.occne-infra \
            -f amf-stubserver.ocnssf2510 \
            -g 80 \
            -h 2 \
            -i ocnssf-nsdb.ocnssf \
            -j ocnssf-db-creds \
            -k ocnssfats-nsconfig.ocnssf2510 \
            -l nrf-stubserver.ocnssf2510 \
            -m ocnssfats \
            -n ocats \
            -o 8081 \
            -p 8080 \
            -q 8080 \
            -r ocnssfats-nsselection.ocnssf2510 \
            -s ocnssfats-nsauditor.ocnssf2510 \
            -t ocnssfats-nsavailability.ocnssf2510 \
            -u ocnssfats-nssubscription.ocnssf2510 \
            -v ocnssfats-ocnssf-app-info.ocnssf2510 \
            -w ocnssfats-ocnssf-perf-info.ocnssf2510 \
            -x 9faf1bbc-6e4a-4454-a507-aef01a101a20 \
            -y nrf-stubserver1.ocnssf2510 \
            -z nrf-stubserver2.ocnssf2510 \
            -A ocnssfats-ocnssf-nrf-client-nfmanagement.ocnssf2510 \
            -B 311 480 \
        '''
        if(env.Include_Regression && "${Include_Regression}" == "YES"){
            sh '''sh /var/lib/jenkins/common_scripts/
merge_jenkinsfile.sh'''
            load "/var/lib/jenkins/ocnssf_tests/jenkinsData/Jenkinsfile-
Merged"
        }
        else{
            load "/var/lib/jenkins/ocnssf_tests/jenkinsData/Jenkinsfile-
Regression"
        }
        }
        }
}
```

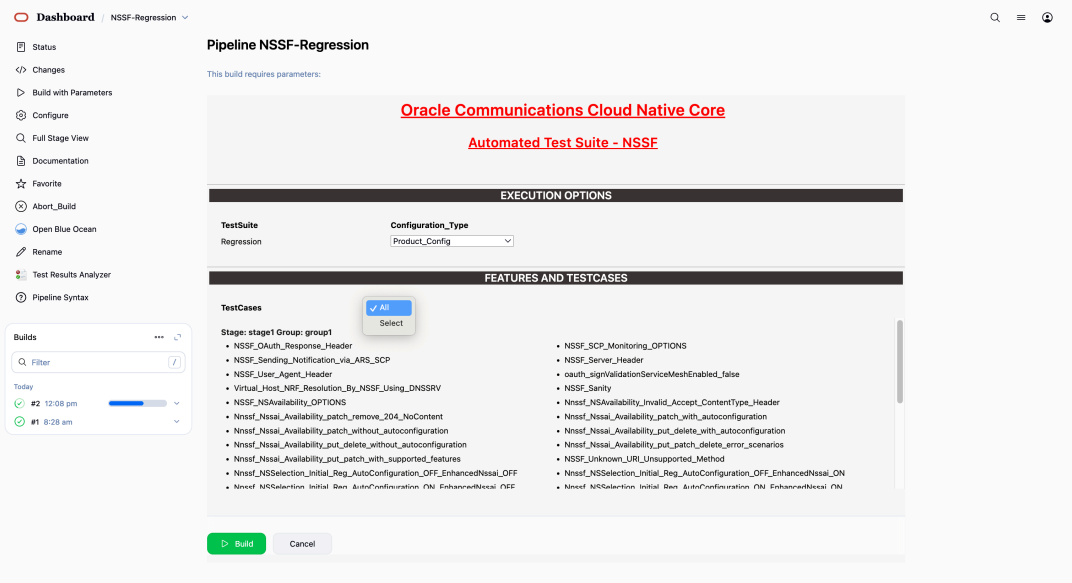
4. Click **Save** after making necessary changes. The **Pipeline NSSF-Regression** screen appears.

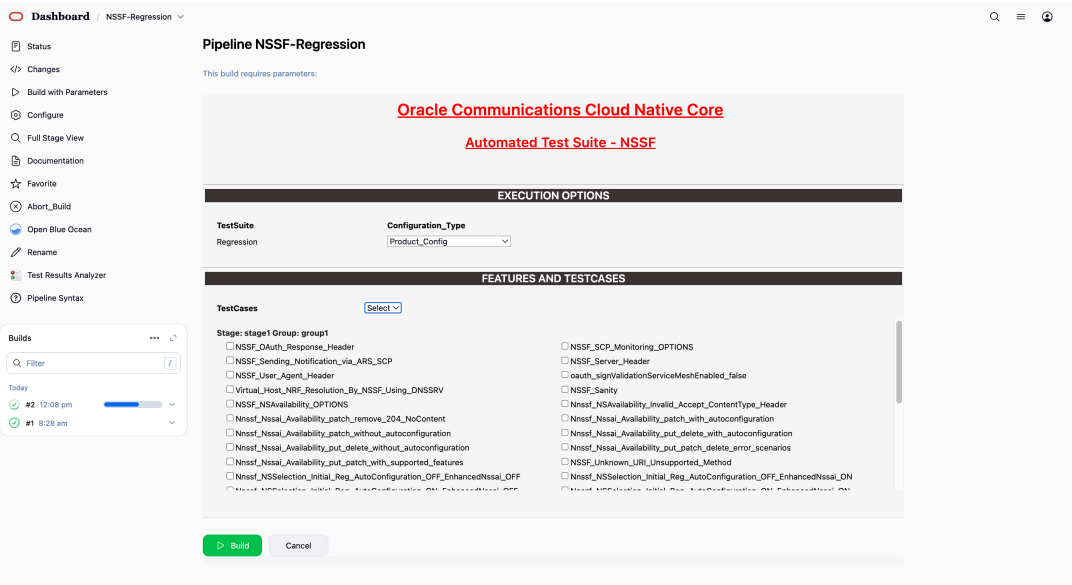
NSSF-Regression - Build with Parameters





- By default, features "ALL," configuration type "Product_config," and include new features "NO" will be selected. All test cases will be executed once you click on BUILD, as shown above.
- Features: ALL. Once you click on ALL, a dropdown "Select" option will appear. After selecting the "Select" option, as shown below, options will appear to select feature files. To run the selected feature files, click on "Build," as shown below.

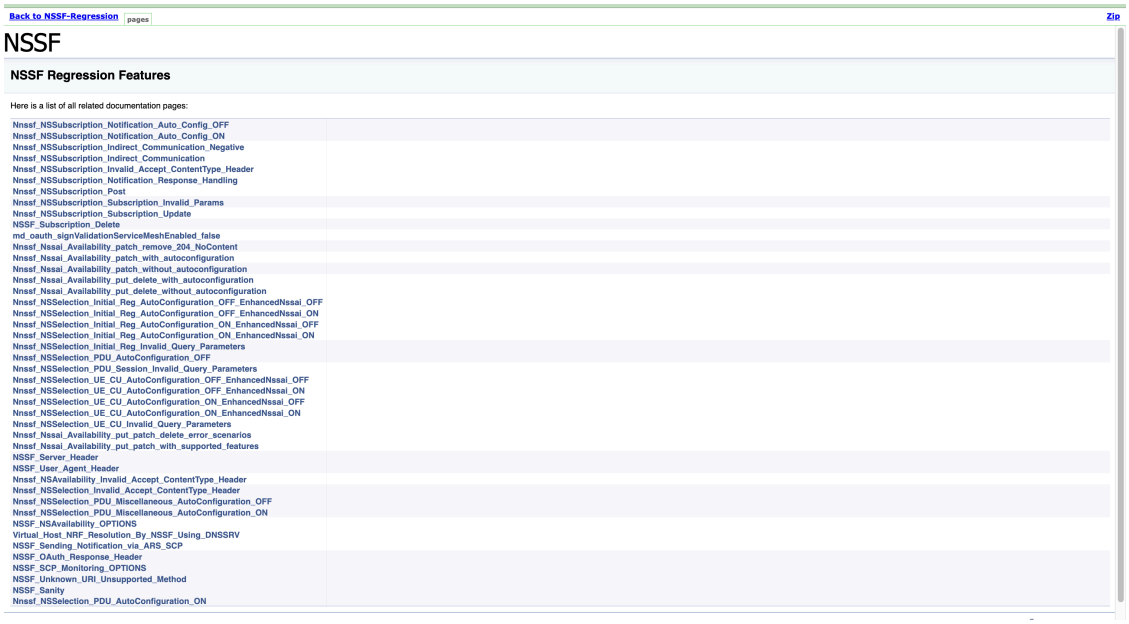




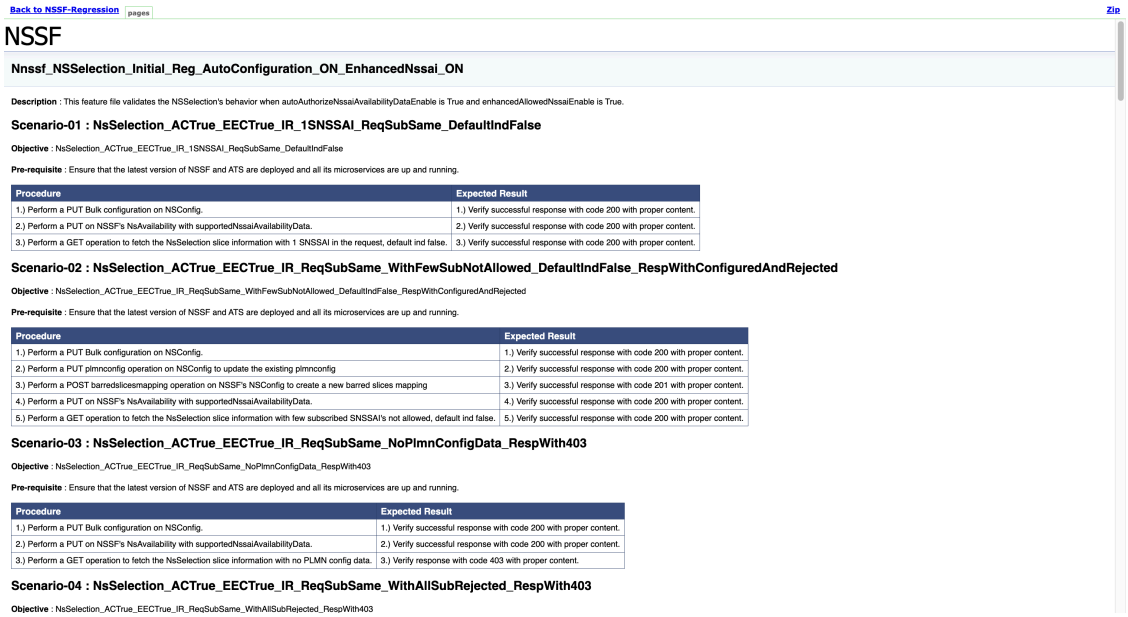

```
45 features passed, 0 failed, 0 skipped
784 scenarios passed, 0 failed, 0 skipped
14570 steps passed, 0 failed, 0 skipped, 0 undefined
Took 196m23.264s
```

4.2.6 NSSF-Regression Documentation

To view NSSF Regression cases, go to **NSSF-Regression** latest pipeline build and click the **Documentation** link in the left navigation pane. The following screen appears. Click any functionality to view its test cases and scenarios of each test case as shown below:



A sample of a few documentation features are as follows:



Once the run of regression features is done successfully, click on the build number whose logs are to be viewed and then click on **Console Output** on left side navigation.

Wait till the page is loaded completely, then click **Download** to download the regression features logs.