vRealize Automation Health Service Extensibility Guide

TECHNICAL WHITE PAPER

MARCH 2018

VERSION 1.0

vmware®

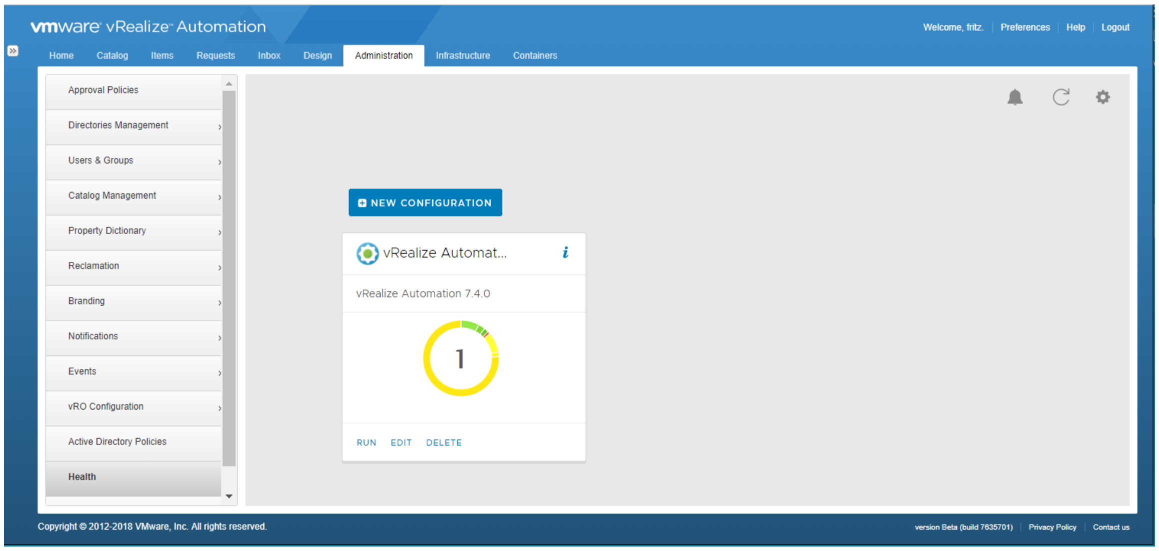
Table of Contents

Revision History

Date	Version	Description
March 2018	1.0	Initial version

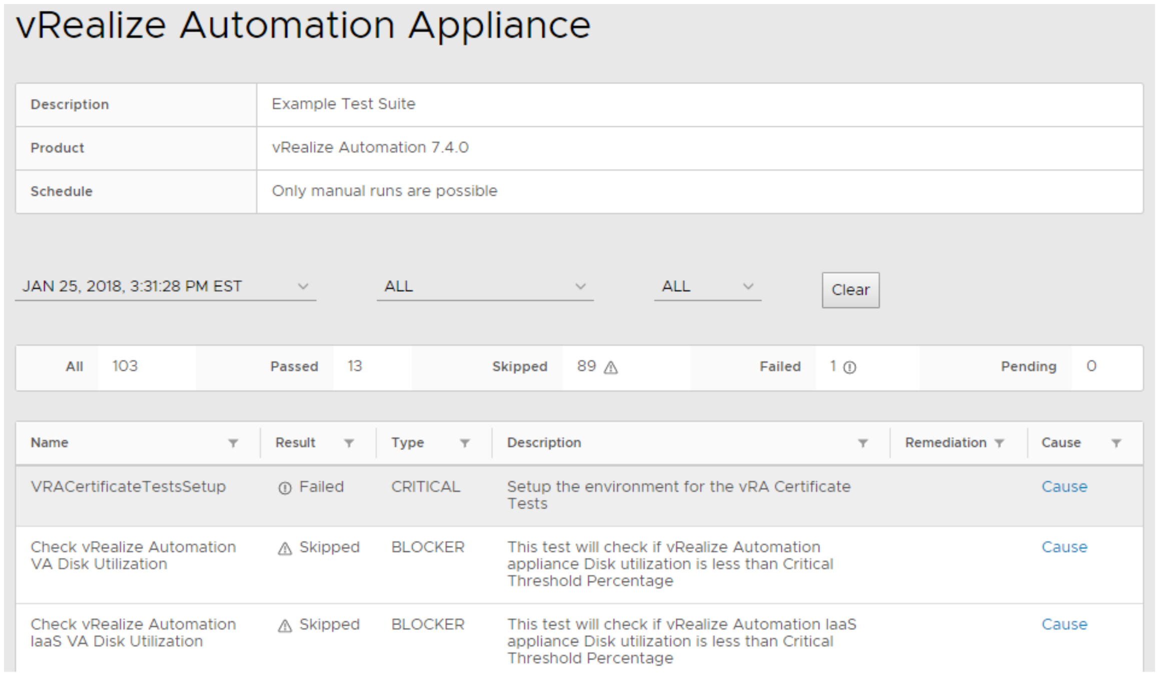
Introduction

You can create custom tests for the vRealize Automation Health Service to assess business-critical functionality. You import your custom tests as single test suite or a collection of test suites and run your tests in the vRealize Automation interface at **Administration** > **Health**.


vmware vRealize Automation
Welcome, fritz. | Preferences | Help | Logout
Home Catalog Items Requests Inbox Design Administration Infrastructure Containers
Approval Policies
Directories Management
Users & Groups
Catalog Management
Property Dictionary
Reclamation
Branding
Notifications
Events
vRO Configuration
Active Directory Policies
Health
NEW CONFIGURATION
vRealize Automat...
vRealize Automation 7.4.0
1
RUN EDIT DELETE
Copyright © 2012-2018 VMware, Inc. All rights reserved.
version Beta (build 7635701) | Privacy Policy | Contact us

Each tile on the Health page represents a configuration. When you create a configuration, you select the test suites that want to run together. When you run your configuration, the consolidated results for your suite appear in a donut diagram.

To see the details about each test in the configuration, click the donut.



vRealize Automation Appliance

Description	Example Test Suite
Product	vRealize Automation 7.4.0
Schedule	Only manual runs are possible

All	Passed	Skipped	Failed	Pending
103	13	89 ⚠	1 ⓘ	0

Name	Result	Type	Description	Remediation	Cause
VRACertificateTestsSetup	ⓘ Failed	CRITICAL	Setup the environment for the vRA Certificate Tests		Cause
Check vRealize Automation VA Disk Utilization	⚠ Skipped	BLOCKER	This test will check if vRealize Automation appliance Disk utilization is less than Critical Threshold Percentage		Cause
Check vRealize Automation IaaS VA Disk Utilization	⚠ Skipped	BLOCKER	This test will check if vRealize Automation IaaS appliance Disk utilization is less than Critical Threshold Percentage		Cause

Creating Custom Tests

Custom tests are written in Python 3, and you should follow best practices for developing Python code so that the test or test suite is easy to maintain.

When custom tests are installed, the system makes a distinction between these concepts:

- **Product**. A product is an application that is part of your vRealize Automation environment. The Health Service provides tests for vRealize Automation or vRealize Orchestrator. You can write custom tests for other products in your environment. When you create a new test configuration, a list of available products appears in the **Product** drop-down menu on the Configuration Details page. Your test suite can define a new product or use an existing product.
- **Test Suite**. A suite is a set of tests for a product. The Health Service provides tests for assessing the health of the system and tenant functionality. When you create a new test configuration, you select one or more tests from a list of available test suites on the Select Test Suites page.

This guide explains how to package, install, and run your custom tests.

Packaging Your Custom Tests

Before you can install your custom tests, you must package your tests with additional files that describe your suite and identify the resources required to run your tests.

For an example of this package, see https://github.com/vmwaresamples/vra-health-samples.

Note: This example uses the pyvmomi package for communicating with vCenter. If your wheel does not need this package, remove it from the install_requires list.

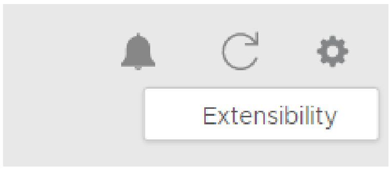
Wheel Setup

Custom tests are packaged using a wheel binary package format. All tests are derived from the TestSuiteBase, which is part of the VMware Health Service SDK. You can download the SDK from vRealize Automation.

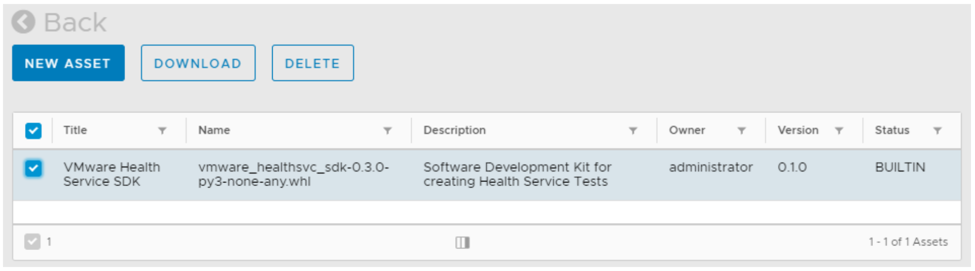
Download the VMware Health Service SDK

Complete this procedure to download the VMware Health Service SDK.

1. On the vRealize Automation interface, click **Administration** > **Health**.
2. Click the cog icon and select **Extensibility**.



3. Select VMware Health Service SDK, and click **Download**.



✓	Title	Name	Description	Owner	Version	Status
✓	VMware Health Service SDK	vmware_healthsvc_sdk-0.3.0-py3-none-any.whl	Software Development Kit for creating Health Service Tests	administrator	0.1.0	BUILTIN

1 1 - 1 of 1 Assets

You arrange the wheel according to this structure:

```
src/
    my_module
           testsuite_my.py
           init.py
    setup.cfg
    setup.py
```

Your file name for the test suite must be begin with `testsuite_` and should suggest what is tested, for example, testsuite_vcenter.py.

Wheel Metadata

You specify the name, version, description, and other wheel elements in the `setup.py` file. This is an example of the `setup.py` file.

```
name='vrealize-health-samples',
  version='1.0.0',
  description='VMware vRealize Health Service Python SDK
Samples',
  author='MyCompany, Inc.',
  author_email='developer@mycompany.com',
  url='http://yourpackageshomepage.com/',
```

Parameter	Description	Example
name	Name you specify for your tests.	`'vrealize-health-samples'`
version	Version of this test.	`'1.0.0'`
description	Description of the test contents.	`'VMware vRealize Health Service Python SDK Samples'`
author	Name of your organization.	`'MyCompany, Inc.'`
author-email	Email address of the test author.	`'developer@mycompany.com'`
url	URL of page describing your tests.	`'http://yourpackageshomepage.com/'`

Required Packages

If your wheel needs additional packages, for example from PIP, the vRealize Health Service requires Internet access to download them.

You specify the packages that you need installed in the `setup.py` file by adding the install_requires parameter and listing the package names in the array. For example,

```
install_requires=['a-package', 'another-package'],
```

Universal Wheel Format

The wheel is organized to create a universal wheel that targets Python 3. The vRealize Automation appliance does not have the resources to build and install platform-specific libraries.
For this reason, you might see an error message like this when you install your wheel in the vRealize Automation appliance:

```
unable to execute '/usr/bin/gcc': No such file or directory
```

This indicates that the system cannot access the required build tools. All the elements in your wheel need to be packaged as a pure Python wheel. For more information, see Python Packaging User Guide.

Test Suite Implementation

The `testsuite_my.py` file in the example package is the implementation of the test suite.

```
from testsuitebase.testsuitebase import test_parameter,
test_suite, TestSuiteBase, test, test_name,
test_description, test_severity, test_access_level,
test_remediation, AccessLevel, Severity

class vCenterTestSuite(TestSuiteBase):

@test_parameter('vra', 'vRealize Automation', 'vaaddress',
'Virtual Appliance Address', 'https://server', 'String')
@test_suite('My External Tests', 'A Bunch of tests',
'vRealize Automation', 7, 4, 0)
```

Test Parameter Annotation: @test_parameter

Position	Parameter	Description	Example
1	Group	All parameters in a group that are displayed together in the vRealize Automation interface.	`'vra'`
2	Group Friendly Name	Friendly name label for the group that appears in the vRealize Automation interface.	`'vRealize Automation'`
3	Parameter	Name of the actual parameter.	`'vaaddress'`

4	Parameter Friendly Name	Friendly name label for the parameter that is referenced in the signature of the test methods.	`'Virtual Appliance Address'`
5	Hint	Appears as text box hint text in the vRealize Automation interface.	`'https://server'`
6	Type	The data type of the parameter for basic validation (String, Password, Integer).	`'String'`

Note: As in this example, all test parameter annotations are defined above the test_suite annotation in the python file.

Test Suite Annotation: @test_suite

Position	**Parameter**	**Description**	**Example**
1	Name	Name of the test suite that appears on the Select Test Suites page during configuration.	`'My External Tests'`
2	Description	Description of the test suite.	`'A Bunch of tests'`
3	Product	Product that the tests target. The product name appears in the Product drop down menu on the Configuration Details page during configuration. All test suites should reference the same product using the same string. Use these names for the vRealize suite products: • 'vRealize Automation' • 'vRealize Orchestrator' • 'vRealize Operations' • 'vRealize Business'	`'vRealize Automation'`
4	Product Version	Product version the tests target. Numbers identify the major, minor, and patch version.	`7, 4, 0`

Test Implementation

A test suite consists of multiple tests. This is an example of a single test.

```
@test_name("Check Username")
@test_description("Checks the value of username.")
@test_access_level(AccessLevel.NORMAL.value)
@test_severity(Severity.NORMAL.value)
@test_remediation("http://www.help.com")
@test
def check_username(self,username):
    assert username == 'root', "expected 'root' but got " +
    username
    return True
```

A test contains these components:

Test Name Annotation: @test_name

@test_name specifies the friendly name of the test, which appears in the Name column of the test report table.

Test Description Annotation: @test_description

@test_description tells what the test does, which appears in the Description column of the test report table.

Test Access Level Annotation: @test_access_level

@test_access_level specifies which user roles can see these test results. Possible values:

Parameter	**Description**
`AccessLevel.NORMAL.value`	Health Consumer users
`AccessLevel.INFRASTRUCTURE.value`	Fabric Administrators
`AccessLevel.ALL.value`	Health Administrators, Infrastructure Administrators

Test Severity Annotation: @test_severity

@test_severity identifies the importance of the test and is used so the test can be sorted in the report. The test severity terms appear in the Type column of the test report table. Possible values:

Parameter	**Description**
`Severity.CRITICAL.value`	If this test fails, other functionality does not work at all.
`Severity.BLOCKER.value`	If this test fails, other functionality is prevented from working.
`Severity.NORMAL.value`	If this test fails, a function might not work normally.

Test Remediation Annotation: @test_remediation

@test-remediation specifies a block of text or an HTTP link to information about how to remediate a failed test. The field defaults to text, but by specifying HTTP, the value is interpreted as an HTTP link. When a test fails, the specified data appears in the Remediation column of the test report table.

Test Annotation: @test

@test is defined last and goes above the test method declaration.

Test Method

This is an example of a test method declaration:

```
def check_username(self,username):
    assert username == 'root', "expected 'root' but got " +
    username
    return True
```

- The first parameter must be `self`.
- Subsequent parameters must match a parameter in `test_parameter` (position 3).
- You can add as many of the defined test parameters as necessary for each test.
- Each test can take a separate set of parameters.

The test should perform one or more asserts. If an assert fails, the test is marked as failed in the Result column of the test report table. The message in the assert displays when you click the Cause link in the Cause column of the test report table.

If the test completes without a failure, it returns True indicating that the test passed.

Sharing Test Parameters

The same parameter names are used in each test for the same product. If you have a test suite which uses the parameter name sqldb_password, another test suite can use sqldb_password as a parameter name. The SQL Database Password field is only displayed once in the UI because all the test suites use the same value. You must declare a parameter name value in your test suite for it to be available for another test suite.

vRealize Automation provides test suites that use these parameter names. Because they are defined, you can use these names in your tests, but you must define the parameter again at the top of your test suite.

vRealize Automation System Test Parameter Names

Parameter Name	**Definition**
url	Address of the vRealize Automation Web Server.
vaDnsName	Address of the vRealize Automation SSH console.
vaUsername	Root user of the vRealize Automation virtual appliance.
ssoAdminUser	SSO administrator user name.
ssoAdminPassword	SSO administrator password.
vaDiskSpaceWarning	Threshold for virtual appliance disk threshold warnings.
vaDiskSpaceCritical	Threshold for virtual appliance disk threshold critical notifications.

vRealize Automation Tenant Test Parameter Names

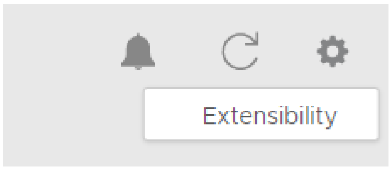
Parameter Name	**Definition**
tenant	Tenant being tested.
tenantUser	Fabric administrator used to run the tests.
tenantPassword	Tenant password.

Uploading Your Test Suite

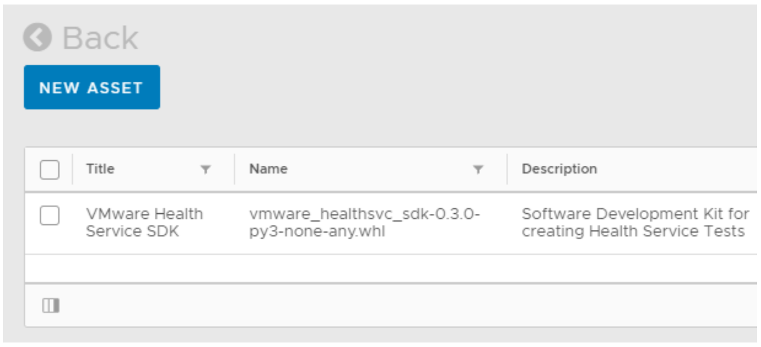
Complete this procedure to upload your test suite.

Prerequisite
Log in to vRealize Automation as IaaS administrator.

1. Click **Administration** > **Health**.
2. In the upper right, click the cog icon and select **Extensibility**.

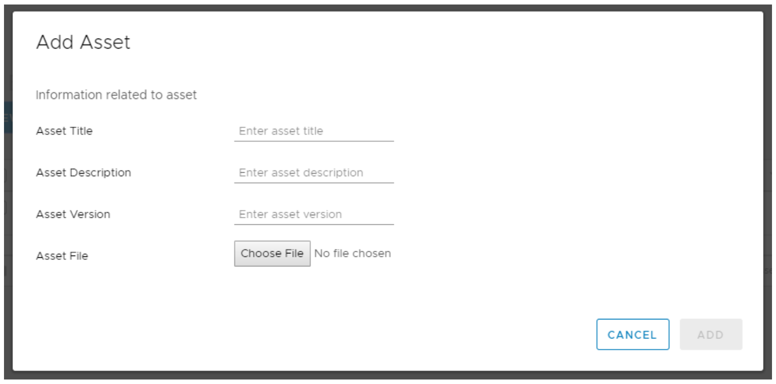


3. Click **New Asset**.



Title	Name	Description
VMware Health Service SDK	vmware_healthsvc_sdk-0.3.0-py3-none-any.whl	Software Development Kit for creating Health Service Tests

4. Enter the requested information about your test suite in the Add Asset box.



Add Asset

Information related to asset

Asset Title	Enter asset title
Asset Description	Enter asset description
Asset Version	Enter asset version
Asset File	Choose File No file chosen

5. Click **Choose File** and select your wheel in your project dist folder.
6. Click **Add**.
   A new row is added to the asset table with the status UPLOADED. When the status changes to INSTALLED, your test is ready to use. If the install process fails, you see a popup that provides a reason.

   Note: If the page does not update, click the refresh icon .

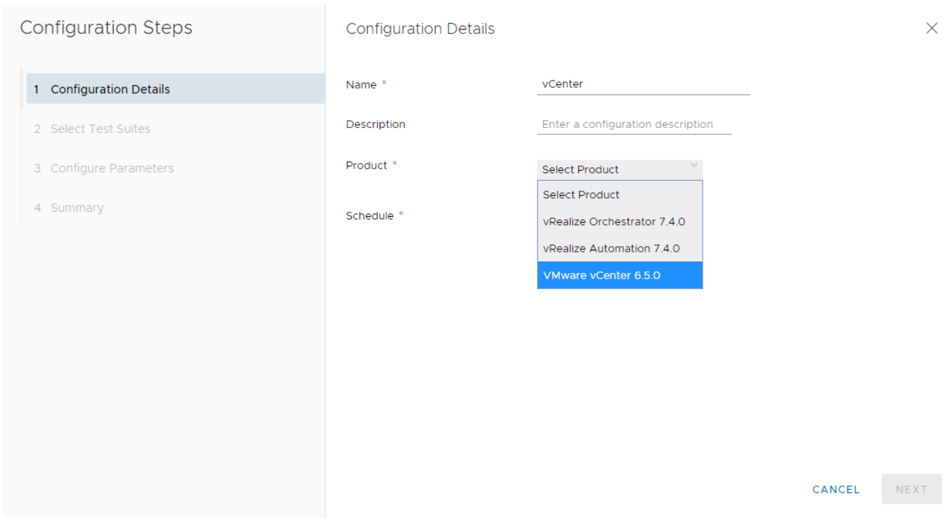
Running Your Test Suite

Complete this procedure to run your test suite.
Note: Installation errors appear as a pop-up message in the interface, so you can investigate the error without searching the log files. Runtime errors are marked as a failed test in the test report, and the error message appears when you click the **Cause** link.

Prerequisite

- Install your test suite.
- Log in to vRealize Automation as IaaS administrator.

1. Click **Administration** > **Health**.
2. Click **New Configuration**.
3. Enter the requested information.



Configuration Steps

1 Configuration Details
2 Select Test Suites
3 Configure Parameters
4 Summary

Configuration Details

Name * vCenter

Description Enter a configuration description

Product * Select Product

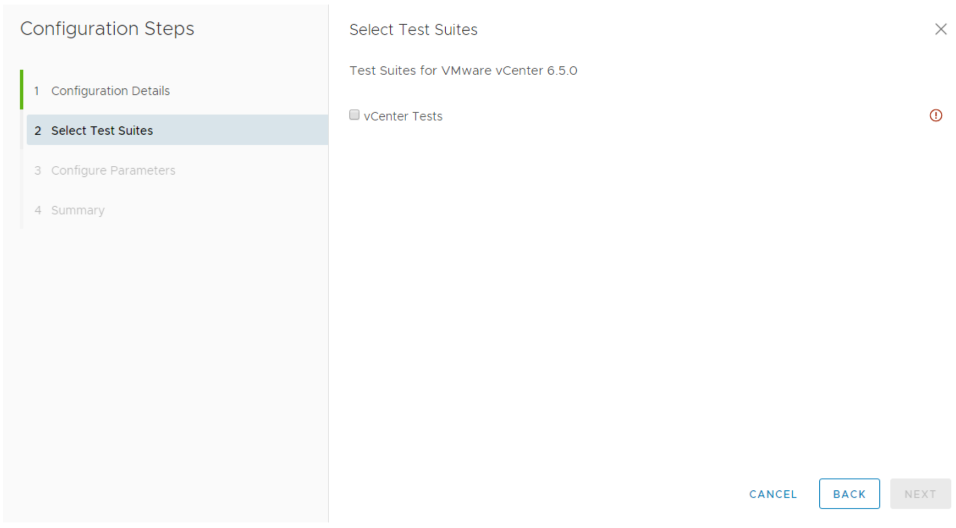
Schedule *

Select Product
vRealize Orchestrator 7.4.0
vRealize Automation 7.4.0
VMware vCenter 6.5.0

CANCEL NEXT

If your test suite adds a new product, select it from the Product drop-down menu. If your test suite does not add a new product, select the product appropriate for your test suite.

4. Click **Next**.
5. Select the check box for your test suite.



Configuration Steps

1 Configuration Details
2 Select Test Suites
3 Configure Parameters
4 Summary

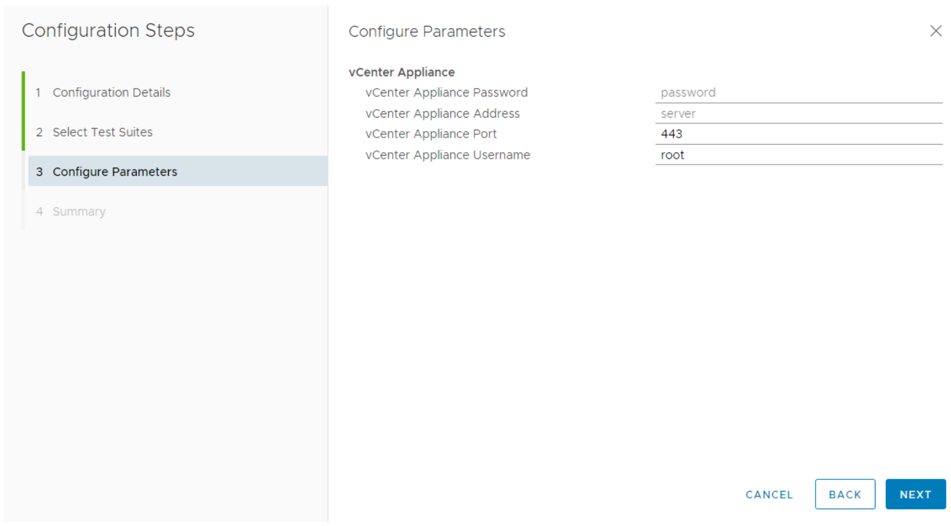
Select Test Suites

Test Suites for VMware vCenter 6.5.0

vCenter Tests

CANCEL BACK NEXT

6. Click **Next**.
7. Enter the parameters you defined in @test_parameters for your test suite.



vCenter Appliance

vCenter Appliance Password	password
vCenter Appliance Address	server
vCenter Appliance Port	443
vCenter Appliance Username	root

8. Click **Next** and **Finish**.
9. On the tile for your test suite, click **Run**.