VIP Validation Report

Deployment: CI Connect 2026.08.1 + WB 2026.08.2+200.pro1 + PM 2026.08.0
Generated: 2026-08-31 18:59:09 UTC

Products Under Test

Product	URL	Version	Results
Connect	`http://localhost:3939`	2026.08.1	7 passed, 2 skipped (9 tests)
Workbench	`http://localhost:8787`	2026.08.2+200.pro1	2 passed (2 tests)
Package Manager	`http://localhost:4242`	2026.08.0	2 passed, 3 skipped (5 tests)

Summary

Total	25
Passed	19
Failed	1
Skipped	5
Status	**FAIL**

Provenance

VIP version	2026.8.3
Run duration	53.5s
Python	3.14.7
Platform	Linux-6.17.0-1022-azure-x86_64-with-glibc2.39
Mode	full
Exit status	1 (tests were collected and run, but some failed)

Failures & Skips

Every check that did not pass, in full. Passing checks are counted above, and every check appears in Detailed Results below.

Failed (1)

FAIL **This check intentionally fails to demonstrate failure rendering** Prerequisites

As someone evaluating a VIP report

`src/vip_tests/prerequisites/test_expected_failure.py::test_intentional_failure  0.00s`

Test procedure

```
› Given a check that is written to fail on purpose
› When the check runs as part of this example report
› Then it fails by design, not because anything is actually broken
```

```
test_intentional_failure: This failure is intentional: it exists only to show how a failed check renders in this example report.
No product configuration will resolve it.
assert 'fail (by design)' == 'pass'
- pass
+ fail (by design)
```

```
fixturefunc = <function check_fails_by_design at 0x7fc87fe20bf0>
request = <FixtureRequest for <Function test_intentional_failure>>
kwargs = {'demo_outcome': {'expected': 'pass', 'actual': 'fail (by design)'}}

    def call_fixture_func(
        fixturefunc: _FixtureFunc[FixtureValue], request: FixtureRequest, kwargs
    ) -> FixtureValue:
        if inspect.isgeneratorfunction(fixturefunc):
            fixturefunc = cast(Callable[..., Generator[FixtureValue]], fixturefunc)
            generator = fixturefunc(**kwargs)
            try:
                fixture_result = next(generator)
            except StopIteration:
                raise ValueError(f"{request.fixturename} did not yield a value") from None
            finalizer = functools.partial(_teardown_yield_fixture, fixturefunc, generator)
            request.addfinalizer(finalizer)
        else:
            fixturefunc = cast(Callable[..., FixtureValue], fixturefunc)
>           fixture_result = fixturefunc(**kwargs)
                             ^^^^^^^^^^^^^^^^^^^^

.venv/lib/python3.14/site-packages/_pytest/fixtures.py:1005:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

demo_outcome = {'expected': 'pass', 'actual': 'fail (by design)'}

    @then("it fails by design, not because anything is actually broken")
    def check_fails_by_design(demo_outcome):
>       assert demo_outcome["actual"] == demo_outcome["expected"], (
            "This failure is intentional: it exists only to show how a failed check "
            "renders in this example report. No product configuration will resolve it."
        )
E       AssertionError: This failure is intentional: it exists only to show how a failed check
    renders in this example report. No product configuration will resolve it.
E       assert 'fail (by design)' == 'pass'
E
E         - pass
E         + fail (by design)

src/vip_tests/prerequisites/test_expected_failure.py:49: AssertionError
```

Likely causes:
- This check is designed to always fail, so the example report has a failure card to show
- No product or configuration problem is involved

Suggested next steps:
1. No action needed. This failure is intentional and does not indicate a real problem
2. It only runs when VIP_ENABLE_EXPECTED_FAILURE_DEMO=1 is set

Documentation: https://github.com/posit-dev/vip/issues/73

Skipped (5)

SKIP **Expected R versions are available on Connect** Connect

No expected R versions specified in vip.toml [runtimes]

`src/vip_tests/connect/test_runtime_versions.py::test_r_versions  0.22s`

Test procedure

```
› Given Connect is accessible at the configured URL
› And expected R versions are specified in vip.toml
› When I query Connect for available R versions
› Then all expected R versions are present
```

SKIP **Expected Python versions are available on Connect** Connect

No expected Python versions specified in vip.toml [runtimes]

`src/vip_tests/connect/test_runtime_versions.py::test_python_versions  0.22s`

Test procedure

```
› Given Connect is accessible at the configured URL
› And expected Python versions are specified in vip.toml
› When I query Connect for available Python versions
› Then all expected Python versions are present
```

SKIP **PyPI mirror is accessible** Package Manager

PyPI package 'requests' not available in any of ['pypi-all'] — repo may not be synced yet

src/vip_tests/package_manager/test_repos.py::test_pypi_mirror 0.02s

Test procedure

```
› Given Package Manager is running
› When I query the PyPI repository for the "requests" package
› Then the package is found in the repository
```

SKIP **Bioconductor mirror is accessible** Package Manager

No Bioconductor repository configured in Package Manager

src/vip_tests/package_manager/test_repos.py::test_bioconductor_mirror 0.01s

Test procedure

```
› Given Package Manager is running
› When I query the Bioconductor repository for the "BiocGenerics" package
› Then the package is found in the repository
```

SKIP **OpenVSX mirror is accessible** Package Manager

No OpenVSX repository configured in Package Manager

src/vip_tests/package_manager/test_repos.py::test_openvsx_mirror 0.01s

Test procedure

```
› Given Package Manager is running
› When I query the OpenVSX repository for the "golang.Go" extension
› Then the package is found in the repository
```

Detailed Results

Connect

9 tests — 7 passed, 2 skipped

PASS **User can log in via the web UI** Connect

`src/vip_tests/connect/test_auth.py::test_connect_login_ui` 1.61s

Test procedure

```
› Given Connect is accessible at the configured URL
› When a user navigates to the Connect login page
› And enters valid credentials
› Then the user is successfully authenticated
› And the Connect dashboard is displayed
```

PASS **API key authentication works** Connect

`src/vip_tests/connect/test_auth.py::test_connect_login_api` 0.45s

Test procedure

```
› Given Connect is accessible at the configured URL
› And a valid API key is configured
› When I request the current user via the API
› Then the API returns user information
```

PASS **Admin user exists and has admin privileges** Connect

`src/vip_tests/connect/test_users.py::test_admin_user` 0.45s

Test procedure

```
› Given Connect is accessible at the configured URL
› When I retrieve the current user profile
› Then the user has admin privileges
```

PASS **Users can be listed** Connect

`src/vip_tests/connect/test_users.py::test_list_users` 0.45s

Test procedure

```
› Given Connect is accessible at the configured URL
› When I list all users
› Then the user list is not empty
› And the test user exists in the user list
```

PASS **Groups can be listed** Connect

`src/vip_tests/connect/test_users.py::test_list_groups` 0.45s

Test procedure

```
› Given Connect is accessible at the configured URL
› When I list all groups
› Then the response is successful
```

SKIP **Expected R versions are available on Connect** Connect

No expected R versions specified in vip.toml [runtimes]

`src/vip_tests/connect/test_runtime_versions.py::test_r_versions` 0.22s

Test procedure

```
› Given Connect is accessible at the configured URL
› And expected R versions are specified in vip.toml
› When I query Connect for available R versions
› Then all expected R versions are present
```

SKIP **Expected Python versions are available on Connect** Connect

No expected Python versions specified in vip.toml [runtimes]

`src/vip_tests/connect/test_runtime_versions.py::test_python_versions` 0.22s

Test procedure

```
› Given Connect is accessible at the configured URL
› And expected Python versions are specified in vip.toml
› When I query Connect for available Python versions
› Then all expected Python versions are present
```

PASS **Expected Quarto versions are available** Connect

src/vip_tests/connect/test_runtime_versions.py::test_quarto_versions 0.45s

Test procedure
› Given Connect is accessible at the configured URL
› When I query Connect for available Quarto versions
› Then at least one Quarto version is available

PASS **Connect system checks can be run and the report downloaded** Connect
src/vip_tests/connect/test_system_checks.py::test_connect_system_checks 29.91s

Test procedure
› Given Connect is accessible at the configured URL
› And a valid API key is configured
› When I trigger a new system check run via the Connect API
› Then the system check report is returned
› And I can download the system check report artifact

Package Manager

5 tests — 2 passed, 3 skipped

SKIP **PyPI mirror is accessible** Package Manager
PyPI package 'requests' not available in any of ['pypi-all'] — repo may not be synced yet
src/vip_tests/package_manager/test_repos.py::test_pypi_mirror 0.02s

Test procedure
› Given Package Manager is running
› When I query the PyPI repository for the "requests" package
› Then the package is found in the repository

SKIP **Bioconductor mirror is accessible** Package Manager
No Bioconductor repository configured in Package Manager
src/vip_tests/package_manager/test_repos.py::test_bioconductor_mirror 0.01s

Test procedure
› Given Package Manager is running
› When I query the Bioconductor repository for the "BiocGenerics" package
› Then the package is found in the repository

SKIP **OpenVSX mirror is accessible** Package Manager
No OpenVSX repository configured in Package Manager
src/vip_tests/package_manager/test_repos.py::test_openvsx_mirror 0.01s

Test procedure
› Given Package Manager is running
› When I query the OpenVSX repository for the "golang.Go" extension
› Then the package is found in the repository

PASS **At least one repository is configured** Package Manager
src/vip_tests/package_manager/test_repos.py::test_repo_exists 0.01s

Test procedure
› Given Package Manager is running
› When I list all repositories
› Then at least one repository exists

PASS **CRAN mirror is accessible** Package Manager
src/vip_tests/package_manager/test_repos.py::test_cran_mirror 1.13s

Test procedure
› Given Package Manager is running
› When I query the CRAN repository for the "Matrix" package
› Then the package is found in the repository

Prerequisites
6 tests — 5 passed, 1 failed

PASS **Workbench server is reachable** Prerequisites
`src/vip_tests/prerequisites/test_components.py::test_product_server_is_reachable[Workbench]`
0.12s

Test procedure
› Given <product> is configured in vip.toml
› When I request the <product> health endpoint
› Then the server responds with a successful status code

PASS **Package Manager server is reachable** Prerequisites
`src/vip_tests/prerequisites/test_components.py::test_product_server_is_reachable[Package Manager]` 0.02s

Test procedure
› Given <product> is configured in vip.toml
› When I request the <product> health endpoint
› Then the server responds with a successful status code

FAIL **This check intentionally fails to demonstrate failure rendering** Prerequisites
As someone evaluating a VIP report
`src/vip_tests/prerequisites/test_expected_failure.py::test_intentional_failure` 0.00s

Test procedure
› Given a check that is written to fail on purpose
› When the check runs as part of this example report
› Then it fails by design, not because anything is actually broken

test_intentional_failure: This failure is intentional: it exists only to show how a failed check renders in this example report. No product configuration will resolve it.
assert 'fail (by design)' == 'pass'
- pass
+ fail (by design)

```
fixturefunc = <function check_fails_by_design at 0x7fc87fe20bf0>
request = <FixtureRequest for <Function test_intentional_failure>>
kwargs = {'demo_outcome': {'expected': 'pass', 'actual': 'fail (by design)'}}

    def call_fixture_func(
        fixturefunc: _FixtureFunc[FixtureValue], request: FixtureRequest, kwargs
    ) -> FixtureValue:
        if inspect.isgeneratorfunction(fixturefunc):
            fixturefunc = cast(Callable[..., Generator[FixtureValue]], fixturefunc)
            generator = fixturefunc(**kwargs)
            try:
                fixture_result = next(generator)
            except StopIteration:
                raise ValueError(f"{request.fixturename} did not yield a value") from None
            finalizer = functools.partial(_teardown_yield_fixture, fixturefunc, generator)
            request.addfinalizer(finalizer)
        else:
            fixturefunc = cast(Callable[..., FixtureValue], fixturefunc)
>           fixture_result = fixturefunc(**kwargs)
                             ^^^^^^^^^^^^^^^^^^^^

.venv/lib/python3.14/site-packages/_pytest/fixtures.py:1005:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

demo_outcome = {'expected': 'pass', 'actual': 'fail (by design)'}

    @then("it fails by design, not because anything is actually broken")
    def check_fails_by_design(demo_outcome):
>       assert demo_outcome["actual"] == demo_outcome["expected"], (
            "This failure is intentional: it exists only to show how a failed check "
            "renders in this example report. No product configuration will resolve it."
        )
E       AssertionError: This failure is intentional: it exists only to show how a failed check
    renders in this example report. No product configuration will resolve it.
E       assert 'fail (by design)' == 'pass'
```

```
E
E         - pass
E         + fail (by design)

src/vip_tests/prerequisites/test_expected_failure.py:49: AssertionError
```

Likely causes:
- This check is designed to always fail, so the example report has a failure card to show
- No product or configuration problem is involved

Suggested next steps:
1. No action needed. This failure is intentional and does not indicate a real problem
2. It only runs when VIP_ENABLE_EXPECTED_FAILURE_DEMO=1 is set

Documentation: https://github.com/posit-dev/vip/issues/73

PASS **Connect version matches configuration** Prerequisites

src/vip_tests/prerequisites/test_versions.py::test_connect_version 0.22s

Test procedure
```
› Given Connect is configured in vip.toml with a version expectation
› When I fetch the Connect server version
› Then the Connect version matches the configured value
```

PASS **Package Manager version matches configuration** Prerequisites

src/vip_tests/prerequisites/test_versions.py::test_package_manager_version 0.02s

Test procedure
```
› Given Package Manager is configured in vip.toml with a version expectation
› When I fetch the Package Manager server version
› Then the Package Manager version matches the configured value
```

PASS **Connect server is reachable** Prerequisites

src/vip_tests/prerequisites/test_components.py::test_product_server_is_reachable[Connect] 0.01s

Test procedure
```
› Given <product> is configured in vip.toml
› When I request the <product> health endpoint
› Then the server responds with a successful status code
```

Security

3 tests — 3 passed

PASS **Unauthenticated API request returns 401** Security

src/vip_tests/security/test_error_handling.py::test_unauthenticated_api_request_returns_401 0.01s

Test procedure
```
› Given Connect is configured in vip.toml
› When I make an unauthenticated API request to Connect
› Then the response status is 401
```

PASS **Invalid API key returns 401** Security

src/vip_tests/security/test_error_handling.py::test_invalid_api_key_returns_401 3.83s

Test procedure
```
› Given Connect is configured in vip.toml
› When I make an API request to Connect with an invalid key
› Then the response status is 401
```

PASS **Non-existent endpoint returns 404** Security

src/vip_tests/security/test_error_handling.py::test_nonexistent_endpoint_returns_404 0.01s

Test procedure
```
› Given Connect is configured in vip.toml
› When I request a non-existent endpoint on Connect
› Then the response status is 404
```

Workbench

2 tests — 2 passed

PASS **User can sign out of Workbench** Workbench

`src/vip_tests/workbench/test_auth.py::test_workbench_signout[chromium]  0.77s`

Test procedure

› Given Workbench is accessible and I am logged in
› When I sign out of Workbench
› Then I am redirected to the Workbench login page

PASS **User can log in to Workbench via the web UI** Workbench

`src/vip_tests/workbench/test_auth.py::test_workbench_login[chromium]  5.23s`

Test procedure

› Given Workbench is accessible at the configured URL
› When a user navigates to the Workbench login page and enters valid credentials
› Then the Workbench homepage is displayed
› And the current user element is visible and non-empty in the header