

VIP Validation Report

Deployment: CI Connect 2026.08.1 + WB 2026.08.2+200.pro1 + PM 2026.08.0
Generated: 2026-09-03 22:34:52 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	1 passed, 1 failed (2 tests)
Package Manager	http://localhost:4242	2026.08.0	2 passed, 3 skipped (5 tests)

Summary

Total	25
Passed	18
Failed	2
Skipped	5
Status	FAIL

Provenance

VIP version	2026.9.0
Run duration	40.8s
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 (2)

FAIL

User can log in to Workbench via the web UI

Workbench

src/vip_tests/workbench/test_auth.py::test_workbench_login[chromium] 5.00s

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

test_workbench_login[chromium]: Login failed: Error: Temporary server error, please try again

```
fixturefunc = <function navigate_and_login at 0x7f9dcd5d12d0>
request = <FixtureRequest for <Function test_workbench_login[chromium]>>
kwargs = {'page': <Page url='http://localhost:8787/auth-sign-in?appUri=%2F&error=2'>,
          'workbench_url': 'http://localhost:8787', 'test_username': 'testuser', 'test_password':
          'RYucVbREuqtjvxT4'}
```

```

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:
src/vip_tests/workbench/test_auth.py:105: in navigate_and_login
    workbench_login(page, workbench_url, test_username, test_password)
-----

page = <Page url='http://localhost:8787/auth-sign-in?appUri=%2F&error=2'>
workbench_url = 'http://localhost:8787', username = 'testuser'
password = 'RYucVbREuqtjvxT4', auth_provider = 'password'
interactive_auth = False, auth_mode = 'none', workbench_auth_error = None
max_retries = 3, retry_delay = 2.0

def workbench_login(
    page: Page,
    workbench_url: str,
    username: str,
    password: str,
    auth_provider: str = "password",
    interactive_auth: bool = False,
    *,
    auth_mode: str = "none",
    workbench_auth_error: str | None = None,
    max_retries: int = 3,
    retry_delay: float = 2.0,
) -> None:
    """Navigate to Workbench homepage, logging in only if required.

    This function:
    - Navigates directly to Workbench's URL
    - Handles OIDC/SSO via pre-loaded storage state (--interactive-auth / --headless-auth)
    - Only fills login form for password auth
    - Retries on transient server errors (e.g., too many logins)

    Args:
        page: Playwright page object
        workbench_url: Base URL for Workbench (e.g., http://localhost:8787)
        username: Login username
        password: Login password
        auth_provider: Auth type (e.g., "password", "oidc", "saml")
        interactive_auth: True when an auth session is pre-loaded (either
            --interactive-auth or --headless-auth)
        auth_mode: Active auth mode ("interactive", "headless", or "none"),
            used to name the responsible CLI flag in skip messages
        workbench_auth_error: Reason the pre-test auth flow could not
            establish a Workbench session, if known. Quoted in the skip
            message so users see the real cause instead of a guess.
        max_retries: Max login attempts on transient errors (default 3)
        retry_delay: Seconds to wait between retries (default 2.0)

    Raises:
        pytest.skip: For non-password auth without a pre-loaded auth session,
            or when the session's storage state doesn't cover Workbench

```

```

        AssertionError: When password login fails after retries
"""
homepage_logo = page.locator(Homepage.POSIT_LOGO)

# For non-password auth without a pre-loaded auth session, skip immediately
if auth_provider != "password" and not interactive_auth:
    pytest.skip(
        f"Login form not available for auth provider {auth_provider!r}. "
        "Pass --interactive-auth or --headless-auth to pre-load browser storage state."
    )

page.goto(workbench_url)
page.wait_for_load_state("load")

# Fast path: already logged in (common with interactive_auth)?
if homepage_logo.is_visible():
    return

# Check if we landed on a login/IdP page
if _on_login_page(page.url):
    # The sign-in page renders client-side after ``load``; wait once for
    # either the password form's username field or an OIDC "Sign in with ..."
    # button to appear before deciding which flow applies. A short fixed wait
    # on only the SSO button races the render and can misread a slow OIDC
    # sign-in page (e.g. an ``?error=2`` bounce) as a password deployment --
    # which then fails the retry loop with "Login failed after 3 attempts"
    # instead of skipping. Waiting for either control settles that race
    # without penalising real password deployments (the username field
    # appears promptly there).
    try:
        page.locator(f"{LoginPage.USERNAME}, button:has-text('Sign
            in')").first.wait_for(
                state="visible", timeout=TIMEOUT_PAGE_LOAD
            )
    except Exception:
        pass

    # An OIDC sign-in page shows a "Sign in with ..." button and no username
    # field. The "sign in" role-name also matches a password form's submit
    # button, so the *absence* of the username field is what distinguishes a
    # true SSO-only page from a password form.
    sso_button = page.get_by_role("button", name=re.compile(r"sign in",
        re.IGNORECASE)).first
    sso_button_present = sso_button.is_visible()
    # A deployment can also skip its own sign-in page entirely and redirect
    # straight to the IdP, whose markup matches neither probe above (see
    # _external_idp_host). Landing off the Workbench origin is conclusive on
    # its own: no Workbench password form is reachable from there.
    idp_host = _external_idp_host(page.url, workbench_url)
    sso_only = idp_host is not None or (
        sso_button_present and not page.locator(LoginPage.USERNAME).is_visible()
    )

    if interactive_auth and sso_only:
        # Storage state was pre-loaded by --interactive-auth / --headless-auth.
        # Workbench's SSO sign-in page does not auto-redirect to the IdP; it renders a
        # "Sign in with OpenID" button. Clicking it triggers a silent SSO round-trip
        # using the saved IdP cookies. The round-trip is serialized across xdist
        # workers (see _silent_sso_signin / oidc_login_lock) to avoid storming the
        # shared IdP session (#484/#467). When the deployment redirected us
        # off-origin instead there is no such button to click, so go straight
        # to the skip -- clicking a locator that resolves to nothing would
        # raise a Playwright error in place of an actionable skip.
        if sso_button_present and _silent_sso_signin(sso_button, homepage_logo,
            workbench_url):
            return # Silent SSO succeeded
        # No usable IdP session (expired, or storage state was stripped for the
        # password-login test) – skip gracefully. Recompute the IdP host: the
        # click above can navigate off-origin before timing out, so where we

```

```

        # ended up is only knowable now, not before the attempt.
        _skip_workbench_session_unproven(
            auth_mode=auth_mode,
            workbench_auth_error=workbench_auth_error,
            landed_url=page.url,
            idp_host=_external_idp_host(page.url, workbench_url),
        )

    if auth_provider != "password":
        _skip_workbench_session_unproven(
            auth_mode=auth_mode,
            workbench_auth_error=workbench_auth_error,
            landed_url=page.url,
            idp_host=_external_idp_host(page.url, workbench_url),
        )
    # Even when auth_provider is reported as "password", the deployment may
    # actually present an SSO/OIDC sign-in page (a "Sign in with ..." button
    # and no username field) – e.g. auth_provider defaulted to "password" on
    # a config-less run against an OIDC deployment. The password login form
    # is unavailable there, so skip rather than fail the retry loop.
    if sso_only:
        where = (
            f"redirects sign-in to the identity provider at {idp_host}"
            if idp_host
            else "presents an SSO/OIDC sign-in page instead (no username/password
                fields)"
        )
        pytest.skip(
            "This scenario requires a Workbench password-login form, but the "
            f"deployment {where}. Password login cannot be exercised on an SSO "
            "deployment, so this scenario is skipped."
        )
    # Password auth - proceed with form login below
else:
    # Not on homepage, not on login page - unexpected state
    # Give it one more check in case page is still loading
    try:
        homepage_logo.wait_for(state="visible", timeout=TIMEOUT_QUICK)
        return
    except Exception:
        pass

# Password authentication with retry logic
login_form = page.locator(LoginPage.USERNAME)
error_panel = page.locator(LoginPage.ERROR_PANEL)

for attempt in range(max_retries):
    if attempt > 0:
        time.sleep(retry_delay)
        page.goto(workbench_url)

    # Fast path check on retry
    if homepage_logo.is_visible():
        return

    # Wait for login form to be ready
    try:
        login_form.wait_for(state="visible", timeout=TIMEOUT_QUICK)
    except Exception:
        continue

    # Fill and submit
    page.fill(LoginPage.USERNAME, username)
    page.fill(LoginPage.PASSWORD, password)

    stay_signed_in = page.locator(LoginPage.STAY_SIGNED_IN)
    if stay_signed_in.is_visible() and not stay_signed_in.is_checked():
        stay_signed_in.click()

```

```

page.click(LoginPage.BUTTON)

# Wait for either homepage (success) or error panel (failure)
homepage_or_error = homepage_logo.or_(error_panel)
try:
    homepage_or_error.wait_for(state="visible", timeout=TIMEOUT_PAGE_LOAD)
except Exception:
    if attempt == max_retries - 1:
        raise AssertionError(f"Login failed after {max_retries} attempts: no response")
    continue

# Check which one appeared
if homepage_logo.is_visible():
    return # Success!

# Error appeared - extract message and maybe retry
if attempt == max_retries - 1:
    error_text = page.locator(LoginPage.ERROR_TEXT).text_content()
    raise AssertionError(f"Login failed: {error_text or 'Unknown error'}")
>
E
AssertionError: Login failed: Error: Temporary server error, please try again

src/vip_tests/workbench/conftest.py:937: AssertionError

```

Likely causes:

- Incorrect test credentials in vip.toml or VIP_TEST_USERNAME/VIP_TEST_PASSWORD
- Auth provider misconfiguration (SAML, OIDC, LDAP, PAM)
- Workbench login page markup changed or is unreachable

Suggested next steps:

1. Verify credentials work by logging in manually at the Workbench URL
2. Check the auth provider configuration in rserver.conf / auth-pam or auth-saml settings
3. If using an external IdP, try `--interactive-auth` or `--headless-auth` instead of a password test

Documentation: https://docs.posit.co/ide/server-pro/authenticating_users/authenticating_users.html

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 0x7f9dcd702400>
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.
        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 0.20s
```

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 0.20s
```

## 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 0.01s
```

## 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_opensvx_mirror 0.02s

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 1.54s

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 0.40s

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 0.40s

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 0.40s

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 0.40s

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 0.20s

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 0.20s

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 0.40s
```

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 29.60s
```

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

PASS CRAN mirror is accessible **Package Manager**

```
src/vip_tests/package_manager/test_repos.py::test_cran_mirror 1.69s
```

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

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 0.01s
```

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_opensvx_mirror 0.02s
```

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

Prerequisites

6 tests — 5 passed, 1 failed

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

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 0x7f9dcd702400>
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 0.20s
```

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 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

Security

3 tests — 3 passed

PASS Invalid API key returns 401 **Security**

```
src/vip_tests/security/test_error_handling.py::test_invalid_api_key_returns_401 2.07s
```

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

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

Workbench

2 tests — 1 passed, 1 failed

PASS User can sign out of Workbench Workbench

src/vip_tests/workbench/test_auth.py::test_workbench_signout[chromium] 0.79s

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

FAIL User can log in to Workbench via the web UI Workbench

src/vip_tests/workbench/test_auth.py::test_workbench_login[chromium] 5.00s

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

test_workbench_login[chromium]: Login failed: Error: Temporary server error, please try again

```
fixturefunc = <function navigate_and_login at 0x7f9dcd5d12d0>
request = <FixtureRequest for <Function test_workbench_login[chromium]>>
kwargs = {'page': <Page url='http://localhost:8787/auth-sign-in?appUri=%2F&error=2'>,
          'workbench_url': 'http://localhost:8787', 'test_username': 'testuser', 'test_password':
          'RYucVbREuqtjvxT4'}

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:
-----
src/vip_tests/workbench/test_auth.py:105: in navigate_and_login
    workbench_login(page, workbench_url, test_username, test_password)
-----

page = <Page url='http://localhost:8787/auth-sign-in?appUri=%2F&error=2'>
workbench_url = 'http://localhost:8787', username = 'testuser'
password = 'RYucVbREuqtjvxT4', auth_provider = 'password'
interactive_auth = False, auth_mode = 'none', workbench_auth_error = None
max_retries = 3, retry_delay = 2.0

def workbench_login(
    page: Page,
    workbench_url: str,
    username: str,
    password: str,
    auth_provider: str = "password",
    interactive_auth: bool = False,
    *,
    auth_mode: str = "none",
    workbench_auth_error: str | None = None,
    max_retries: int = 3,
    retry_delay: float = 2.0,
) -> None:
    """Navigate to Workbench homepage, logging in only if required.
```

```

This function:
- Navigates directly to Workbench's URL
- Handles OIDC/SSO via pre-loaded storage state (--interactive-auth / --headless-auth)
- Only fills login form for password auth
- Retries on transient server errors (e.g., too many logins)

Args:
  page: Playwright page object
  workbench_url: Base URL for Workbench (e.g., http://localhost:8787)
  username: Login username
  password: Login password
  auth_provider: Auth type (e.g., "password", "oidc", "saml")
  interactive_auth: True when an auth session is pre-loaded (either
    --interactive-auth or --headless-auth)
  auth_mode: Active auth mode ("interactive", "headless", or "none"),
    used to name the responsible CLI flag in skip messages
  workbench_auth_error: Reason the pre-test auth flow could not
    establish a Workbench session, if known. Quoted in the skip
    message so users see the real cause instead of a guess.
  max_retries: Max login attempts on transient errors (default 3)
  retry_delay: Seconds to wait between retries (default 2.0)

Raises:
  pytest.skip: For non-password auth without a pre-loaded auth session,
    or when the session's storage state doesn't cover Workbench
  AssertionError: When password login fails after retries
"""
homepage_logo = page.locator(Hompage.POSIT_LOGO)

# For non-password auth without a pre-loaded auth session, skip immediately
if auth_provider != "password" and not interactive_auth:
    pytest.skip(
        f"Login form not available for auth provider {auth_provider!r}. "
        "Pass --interactive-auth or --headless-auth to pre-load browser storage state."
    )

page.goto(workbench_url)
page.wait_for_load_state("load")

# Fast path: already logged in (common with interactive_auth)?
if homepage_logo.is_visible():
    return

# Check if we landed on a login/IdP page
if _on_login_page(page.url):
    # The sign-in page renders client-side after ``load``; wait once for
    # either the password form's username field or an OIDC "Sign in with ..."
    # button to appear before deciding which flow applies. A short fixed wait
    # on only the SSO button races the render and can misread a slow OIDC
    # sign-in page (e.g. an ``?error=2`` bounce) as a password deployment --
    # which then fails the retry loop with "Login failed after 3 attempts"
    # instead of skipping. Waiting for either control settles that race
    # without penalising real password deployments (the username field
    # appears promptly there).
    try:
        page.locator(f"{LoginPage.USERNAME}, button:has-text('Sign
            in')").first.wait_for(
                state="visible", timeout=TIMEOUT_PAGE_LOAD
            )
    except Exception:
        pass

    # An OIDC sign-in page shows a "Sign in with ..." button and no username
    # field. The "sign in" role-name also matches a password form's submit
    # button, so the *absence* of the username field is what distinguishes a
    # true SSO-only page from a password form.
    sso_button = page.get_by_role("button", name=re.compile(r"sign in",
        re.IGNORECASE)).first
    sso_button_present = sso_button.is_visible()

```

```

# A deployment can also skip its own sign-in page entirely and redirect
# straight to the IdP, whose markup matches neither probe above (see
# _external_idp_host). Landing off the Workbench origin is conclusive on
# its own: no Workbench password form is reachable from there.
idp_host = _external_idp_host(page.url, workbench_url)
sso_only = idp_host is not None or (
    sso_button_present and not page.locator(LoginPage.USERNAME).is_visible()
)

if interactive_auth and sso_only:
    # Storage state was pre-loaded by --interactive-auth / --headless-auth.
    # Workbench's SSO sign-in page does not auto-redirect to the IdP; it renders a
    # "Sign in with OpenID" button. Clicking it triggers a silent SSO round-trip
    # using the saved IdP cookies. The round-trip is serialized across xdist
    # workers (see _silent_sso_signin / oidc_login_lock) to avoid storming the
    # shared IdP session (#484/#467). When the deployment redirected us
    # off-origin instead there is no such button to click, so go straight
    # to the skip -- clicking a locator that resolves to nothing would
    # raise a Playwright error in place of an actionable skip.
    if sso_button_present and _silent_sso_signin(sso_button, homepage_logo,
        workbench_url):
        return # Silent SSO succeeded
    # No usable IdP session (expired, or storage state was stripped for the
    # password-login test) – skip gracefully. Recompute the IdP host: the
    # click above can navigate off-origin before timing out, so where we
    # ended up is only knowable now, not before the attempt.
    _skip_workbench_session_unproven(
        auth_mode=auth_mode,
        workbench_auth_error=workbench_auth_error,
        landed_url=page.url,
        idp_host=_external_idp_host(page.url, workbench_url),
    )

if auth_provider != "password":
    _skip_workbench_session_unproven(
        auth_mode=auth_mode,
        workbench_auth_error=workbench_auth_error,
        landed_url=page.url,
        idp_host=_external_idp_host(page.url, workbench_url),
    )

# Even when auth_provider is reported as "password", the deployment may
# actually present an SSO/OIDC sign-in page (a "Sign in with ..." button
# and no username field) – e.g. auth_provider defaulted to "password" on
# a config-less run against an OIDC deployment. The password login form
# is unavailable there, so skip rather than fail the retry loop.
if sso_only:
    where = (
        f"redirects sign-in to the identity provider at {idp_host}"
        if idp_host
        else "presents an SSO/OIDC sign-in page instead (no username/password
            fields)"
    )
    pytest.skip(
        "This scenario requires a Workbench password-login form, but the "
        f"deployment {where}. Password login cannot be exercised on an SSO "
        "deployment, so this scenario is skipped."
    )

# Password auth – proceed with form login below
else:
    # Not on homepage, not on login page – unexpected state
    # Give it one more check in case page is still loading
    try:
        homepage_logo.wait_for(state="visible", timeout=TIMEOUT_QUICK)
        return
    except Exception:
        pass

# Password authentication with retry logic
login_form = page.locator(LoginPage.USERNAME)

```

```

error_panel = page.locator(LoginPage.ERROR_PANEL)

for attempt in range(max_retries):
    if attempt > 0:
        time.sleep(retry_delay)
        page.goto(workbench_url)

    # Fast path check on retry
    if homepage_logo.is_visible():
        return

    # Wait for login form to be ready
    try:
        login_form.wait_for(state="visible", timeout=TIMEOUT_QUICK)
    except Exception:
        continue

    # Fill and submit
    page.fill(LoginPage.USERNAME, username)
    page.fill(LoginPage.PASSWORD, password)

    stay_signed_in = page.locator(LoginPage.STAY_SIGNED_IN)
    if stay_signed_in.is_visible() and not stay_signed_in.is_checked():
        stay_signed_in.click()

    page.click(LoginPage.BUTTON)

    # Wait for either homepage (success) or error panel (failure)
    homepage_or_error = homepage_logo.or_(error_panel)
    try:
        homepage_or_error.wait_for(state="visible", timeout=TIMEOUT_PAGE_LOAD)
    except Exception:
        if attempt == max_retries - 1:
            raise AssertionError(f"Login failed after {max_retries} attempts: no response")
        continue

    # Check which one appeared
    if homepage_logo.is_visible():
        return # Success!

    # Error appeared - extract message and maybe retry
    if attempt == max_retries - 1:
        error_text = page.locator(LoginPage.ERROR_TEXT).text_content()
        raise AssertionError(f"Login failed: {error_text or 'Unknown error'}")
>
E       AssertionError: Login failed: Error: Temporary server error, please try again

src/vip_tests/workbench/conftest.py:937: AssertionError

```

Likely causes:

- Incorrect test credentials in vip.toml or VIP_TEST_USERNAME/VIP_TEST_PASSWORD
- Auth provider misconfiguration (SAML, OIDC, LDAP, PAM)
- Workbench login page markup changed or is unreachable

Suggested next steps:

1. Verify credentials work by logging in manually at the Workbench URL
2. Check the auth provider configuration in rserver.conf / auth-pam or auth-saml settings
3. If using an external IdP, try `--interactive-auth` or `--headless-auth` instead of a password test

Documentation: https://docs.posit.co/ide/server-pro/authenticating_users/authenticating_users.html