Service Accounts - Storybook Implementation Notes

This document describes the process of setting up Storybook with User Journeys for the Service Accounts application. The primary goal is to create automated tests that verify core user flows before and after migrating the table component to PatternFly DataView.

Table of Contents

Overview

Purpose

The Storybook User Journeys serve as:

1. **Regression tests** - Verify that core functionality works before and after DataView migration
2. **Documentation** - Living documentation of user flows
3. **CI checks** - Automated validation on every PR via Chromatic

Architecture

```
.storybook/
├── main.ts              # Storybook configuration with webpack aliases
├── preview.tsx          # Global decorators, providers, MSW setup
└── mocks/
    ├── useChrome.ts     # Mock for @redhat-cloud-services/frontend-components/useChro
    ├── DateFormat.tsx   # Mock for DateFormat component
    ├── Main.tsx         # Mock for Main layout component
    ├── PageHeader.tsx   # Mock for PageHeader component
    └── useNotifications.ts  # Mock for notification hooks

src/Routes/ListServiceAccountsPage/
├── ListServiceAccountsPage.stories.tsx  # User Journeys (Create, Delete, Reset)
└── ServiceAccountsTable.stories.tsx     # Component-level stories
```

Key Configuration

Node.js Version

Storybook 10 requires Node.js 20.19+. The version is specified in `.nvmrc`:

```
20.19.0
```

TypeScript Configuration

The `tsconfig.json` uses `moduleResolution: "bundler"` for Storybook 10 ESM compatibility:

```json
{
  "compilerOptions": {
    "moduleResolution": "bundler",
    "module": "esnext",
    "target": "es5",
    "strict": true,
    "jsx": "react",
    "esModuleInterop": true
  }
}
```

> **Note:** This matches the configuration used by `insights-chrome`, the core platform application.

Storybook Dependencies

```json
{
  "devDependencies": {
    "storybook": "^10.1.11",
    "@storybook/react-webpack5": "^10.1.11",
    "@storybook/addon-docs": "^10.1.11",
    "@storybook/addon-webpack5-compiler-swc": "^4.0.2",
    "@storybook/test-runner": "^0.24.2",
    "msw": "^2.12.7",
    "msw-storybook-addon": "^2.0.6"
  }
}
```

Storybook Main Configuration

The `.storybook/main.ts` uses `process.cwd()` for ESM compatibility:

```
import type { StorybookConfig } from '@storybook/react-webpack5';
import path from 'path';

const storybookDir = path.join(process.cwd(), '.storybook');

const config: StorybookConfig = {
  stories: ['../src/**/*.stories.@(js|jsx|mjs|ts|tsx)'],
  addons: [
    '@storybook/addon-webpack5-compiler-swc',
    'msw-storybook-addon',
    '@storybook/addon-docs'
  ],
  framework: {
    name: '@storybook/react-webpack5',
    options: {}
  },
  staticDirs: ['../public'],
  webpackFinal: async (config) => {
    // Mock frontend-components modules
    config.resolve.alias = {
      ...config.resolve.alias,
      '@redhat-cloud-services/frontend-components/useChrome':
        path.join(storybookDir, 'mocks/useChrome.ts'),
      // ... other mocks
    };
    return config;
  },
};
```

MSW (Mock Service Worker) Setup

MSW is used for API mocking. The `preview.tsx` initializes MSW:

```
import { initialize, mswLoader } from 'msw-storybook-addon';

initialize({ onUnhandledRequest: 'bypass' });

const preview: Preview = {
  loaders: [mswLoader],
  // ...
};
```

The service worker file (`public/mockServiceWorker.js`) is committed to the repository.

Portal Root for Modals

Modals use `appendTo` to render into a portal. A root element is created in `preview.tsx`:

```
if (!document.getElementById('chrome-app-render-root')) {
  const portalRoot = document.createElement('div');
  portalRoot.id = 'chrome-app-render-root';
  document.body.appendChild(portalRoot);
}
```

User Journeys Design

Stateful MSW Handlers

All journeys share stateful MSW handlers that maintain in-memory state:

```
const createStatefulHandlers = (initialServiceAccounts: ServiceAccount[]) => {
  let serviceAccounts = [...initialServiceAccounts];

  return [
    // GET - List
    http.get('*/apis/service_accounts/v1', ({ request }) => {
      const url = new URL(request.url);
      const first = parseInt(url.searchParams.get('first') || '0');
      const max = parseInt(url.searchParams.get('max') || '50');
      return HttpResponse.json(serviceAccounts.slice(first, first + max));
    }),

    // GET - Single (for Delete/Reset modals)
    http.get('*/apis/service_accounts/v1/:id', ({ params }) => {
      const sa = serviceAccounts.find(s => s.id === params.id || s.clientId === params
      return sa ? HttpResponse.json(sa) : new HttpResponse(null, { status: 404 });
    }),

    // POST - Create
    http.post('*/apis/service_accounts/v1', async ({ request }) => {
      const body = await request.json();
      const newSA = { id: `sa-${Date.now()}`, ...body };
      serviceAccounts = [newSA, ...serviceAccounts];
      return HttpResponse.json(newSA, { status: 201 });
    }),

    // DELETE
    http.delete('*/apis/service_accounts/v1/:id', ({ params }) => {
      serviceAccounts = serviceAccounts.filter(sa => sa.id !== params.id);
      return new HttpResponse(null, { status: 204 });
    }),

    // POST - Reset credentials
    http.post('*/apis/service_accounts/v1/:id/resetSecret', ({ params }) => {
      const sa = serviceAccounts.find(s => s.id === params.id);
      return HttpResponse.json({ ...sa, secret: `new-secret-${Date.now()}` });
    }),
  ];
};
```

Journey 1: Create Service Account

Purpose: Test the complete flow of creating a new service account with validation.

Steps:

Step	Action	Assertion
1	Wait for page load	Service accounts table is visible
2	Verify initial state	All 3 mock accounts displayed
3	Click "Create service account"	Modal opens
4	Enter invalid name (uppercase)	Validation error appears
5	Clear and enter valid name	Validation error disappears
6	Enter description	-
7	Click "Create" button	-
8	Wait for credentials modal	Modal with Client ID and Secret appears
9	Check confirmation checkbox	Close button becomes enabled
10	Click "Close"	Modal closes
11	Verify new account in list	"test-journey-account" is visible
12	Verify pagination	Shows "1 - 4"

Journey 2: Delete Service Account

Purpose: Test the complete flow of deleting a service account.

Steps:

Step	Action	Assertion
1	Wait for page load	Service accounts table is visible
2	Verify initial state	All 3 mock accounts displayed
3	Find "data-pipeline" row	Row exists
4	Click kebab menu (last button in row)	Dropdown opens
5	Click "Delete service account"	-
6	Wait for confirmation modal	Modal shows account name
7	Click "Delete" button	-
8	Wait for modal to close	-
9	Verify account removed	"data-pipeline" not in list

Step	Action	Assertion
10	Verify remaining accounts	Other accounts still present
11	Verify pagination	Shows "1 - 2"

Journey 3: Reset Service Account Credentials

Purpose: Test the complete flow of resetting credentials.

Steps:

Step	Action	Assertion
1	Wait for page load	Service accounts table is visible
2	Verify initial state	Service accounts displayed
3	Find "my-api-service" row	Row exists
4	Click kebab menu	Dropdown opens
5	Click "Reset credentials"	-
6	Wait for confirmation modal	Modal shows account name
7	Click "Reset" button	-
8	Wait for credentials modal	New credentials displayed
9	Check confirmation checkbox	-
10	Click "Close"	Modal closes
11	Verify account still exists	Account not deleted

Problems Encountered and Solutions

1. Node.js Version Incompatibility

Problem: Storybook 10 requires Node.js 20.19+, but the system had v20.12.2.

Solution:

- Created `.nvmrc` with `20.19.0`

- Use `nvm use` before running Storybook commands
- CI workflow uses the `.nvmrc` file automatically

2. ESM Module Resolution

Problem: TypeScript couldn't find Storybook 10 modules with `moduleResolution: "node"`.

Error:

```
Cannot find module '@storybook/react-webpack5' or its corresponding type declarations.
```

Solution: Changed `tsconfig.json` to use `moduleResolution: "bundler"`, which is also used by `insights-chrome`.

3. `__dirname` Not Defined in ESM

Problem: Storybook 10 runs config files as ESM, where `__dirname` is not available.

Error:

```
ReferenceError: __dirname is not defined
```

Solution: Use `process.cwd()` instead:

```
const storybookDir = path.join(process.cwd(), '.storybook');
```

4. Modal Portals Not Rendering

Problem: Modals use `appendTo` to render into `#chrome-app-render-root`, which doesn't exist in Storybook.

Solution: Create the portal root element in `preview.tsx`:

```
if (!document.getElementById('chrome-app-render-root')) {
  const portalRoot = document.createElement('div');
  portalRoot.id = 'chrome-app-render-root';
  document.body.appendChild(portalRoot);
}
```

5. Router Conflicts

Problem: Global `MemoryRouter` in preview conflicted with story-specific routers.

Error:

```
You cannot render a <Router> inside another <Router>
```

Solution: Created a self-contained `PageWithRoutes` wrapper component for journey stories that includes its own router, and disabled global decorators via `parameters.skipGlobalDecorators: true`.

6. Finding Modal Elements

Problem: Modal elements are portaled outside the canvas element, so `within(canvasElement)` can't find them.

Solution: Use `document.querySelector` or `document.getElementById` for modal elements:

```
const modalDialog = document.querySelector('[role="dialog"]');
const modal = within(modalDialog);
```

7. Finding Kebab Menu Items

Problem: OUIA selectors weren't reliable for finding menu items.

Solution: Use `within(document.body).findByText()`:

```
const body = within(document.body);
const deleteMenuItem = await body.findByText('Delete service account', {}, { timeout:
```

8. Storybook 9 vs 10 Compatibility

Problem: Initially tried to use Storybook 9 for consistency with `insights-rbac-ui`, but encountered test-runner errors.

Error:

```
ReferenceError: Cannot access 'StorybookTestRunnerError' before initialization
```

Solution: Stayed with Storybook 10, which worked correctly with the proper Node.js version and ESM configuration.

Running Tests

Prerequisites

```
# Use correct Node.js version
nvm use

# Install dependencies
npm install
```

Commands

Command	Description
`npm run storybook`	Start Storybook dev server
`npm run test-storybook`	Run tests (requires Storybook running)
`npm run test-storybook:ci`	Run tests in CI mode
`npm run test-storybook:fast`	Run tests without typing delays

CI Integration

The project includes a Chromatic workflow (`.github/workflows/chromatic.yml`) that:

1. Deploys Storybook to Chromatic
2. Runs Storybook tests against the deployed instance

Migration Considerations

What Should Survive Migration

The journeys test **user-visible behavior**, not implementation details:

- Clicking buttons and links
- Filling forms
- Verifying text content
- Checking pagination

Expected Changes After DataView Migration

1. **Table structure** - DataView uses different CSS classes and HTML structure
2. **Pagination** - DataView may use `useDataViewPagination` hook
3. **Loading states** - May use different skeleton patterns

Recommended Approach

1. Run journeys before migration to ensure baseline
2. Perform DataView migration
3. Run journeys - expect some failures
4. Update selectors and assertions as needed
5. Journeys should pass with minimal changes

Selectors to Watch

Current	May Change To
`.pf-v6-c-table`	DataView table classes
`tbody tr`	DataView row structure
Kebab button (last button in row)	May have OUIA IDs

Summary

This setup provides:

- ✅ **8 passing tests** covering all CRUD operations
- ✅ **Stateful mocking** with MSW for realistic API simulation
- ✅ **CI-ready** with Chromatic integration
- ✅ **Migration-resilient** design focusing on user behavior

The journeys serve as a safety net for the upcoming DataView migration, ensuring core functionality remains intact.