Requirement Doc - Backbone to React (ES6 Compatible) using Report API

Applicable Release: E2 only

Action Owner:

1. Product Development Use-cases:
2. Upgrade Use-cases: &

1. Objective

Migrate all custom report-level Backbone JS scripts (stored in the “additional settings” CLOB) to React (ES6), ensuring compatibility with E2 and integration with the latest Report APIs.

2. Background

- **Legacy Implementation:** In E1 (and earlier), custom logic for reports is implemented in Backbone JS and persisted in the “additional settings” of report metadata.
- **Upgrade Impact:** After upgrading to E2, legacy Backbone code will fail or behave unpredictably.
- **API Enhancement:** E2 has updated Report APIs that must be used for any server-side data interactions.

3. **In-scope:**
    1. All existing reports whose metadata “additional settings” contain Backbone JS.
    2. Conversion of that code to React (ES6 syntax).
    3. Support both UI modes. E2 permits both legacy Backbone and new React render paths. This use case is needed to support.

```
Sample code
if(window.Backbone)
{
  //backbone code
} else {
  // react compliant code
}
```

    4. Integration with latest Report API methods as defined in the Report API guide. This should include the capabilities to
        1. Make changes to existing React JS code for additional requests if any.
        2. Additional requests could be
            1. Change in existing logic
            2. New functionality using Report APIs

3. Write new React compliant code. This request can be from a customer or part of product enhancement.
4. Syntax correction, if any in the existing React converted code.

4. Conversion Scenarios

Scenario	Action
A. No existing Report API calls	1. Rewrite Backbone JS logic as React (ES6) compatible format. 2. Replace JSs with appropriate Report API calls.
B. Contains legacy Report API invocations	1. Refactor existing API calls to match the latest Report API. 2. Rewrite surrounding logic in ES6, using React patterns.

5. Current Team Process

How the team currently performs conversions

1. **Access Report:** Developer navigates to Module → Reports → selects the target report.
2. **Extract Backbone JS:** Opens the Additional Settings panel and copies the existing Backbone script.
3. **Manual Conversion:** Rewrites the script in React (ES6) syntax, following Report API guide and examples.
4. **API Integration:** Refers to the Report API Guide to replace or update JS logics with the latest report APIs and examples.
5. **Deploy & Test:** Pastes converted React code back into Additional Settings, saves the report, and validates behavior.

6. Admin-UX Conversion Agent Expectations

6.1 Content Conversion Workflow

- **JS Input:** Provide a field for developers to paste existing Backbone JS from Additional Settings.
- **Conversion Plan Generation:** Automatically outline a step-by-step plan that includes:
    1. ES6 module structure (imports/exports, components, etc..).
    2. Report APIs for corresponding JS logic. Generate and consider the syntax grammar for Report APIs for JS logic. Having syntax grammar would avoid syntactical validation and it would be first time right.
    3. Placeholder points for any custom UI logic or business rules, based on Developer's additional asks for Business logic.
- **Feedback & Customization:** Allow developers to review and edit the plan or API selections; support inline comments.

- **Additional Logic Integration:** Let developers specify extra UI elements (e.g. buttons or handlers) or additional business logic so the plan and final code include these.
- **Code Generation:** On approval, produce React (ES6) code embedding updated Report API calls and any custom logic.
- **Copy & Deploy:** Enable easy copying of generated code for pasting into Additional Settings and testing.

6.2 Editor Use Case

- **Report Discovery:** UI to browse Modules → Module → Objects → Reports → select target report.
- **Metadata Retrieval:** Read and display current Backbone JS from Additional Settings.
- **Guided Conversion:** Invoke the Content Conversion Workflow with review and approval controls.

6.3 Testing & Validation:

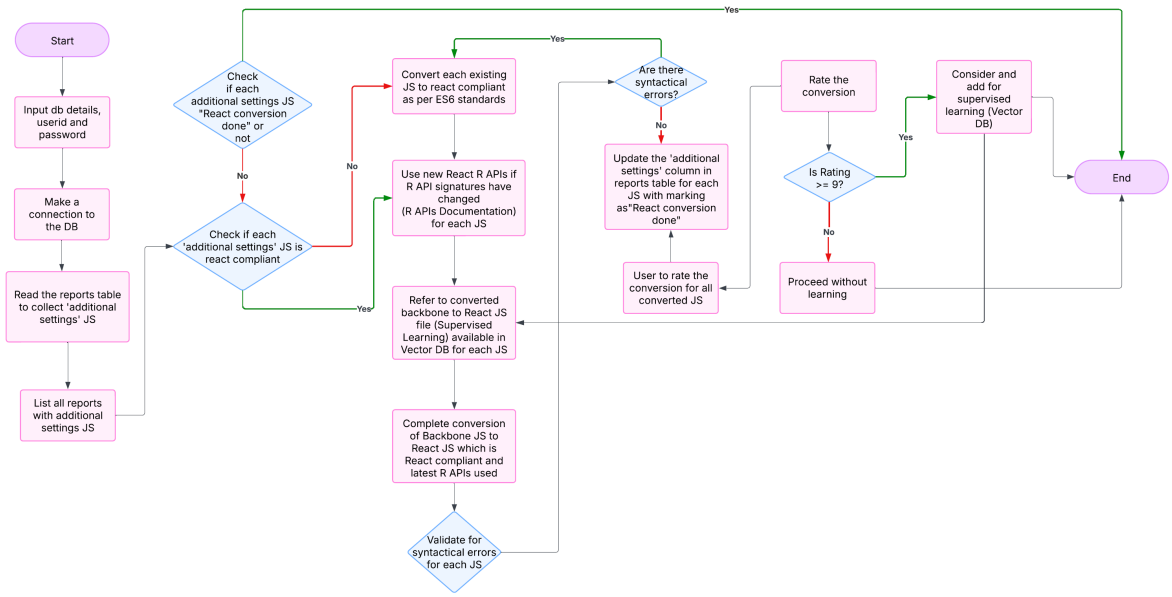
- Verify syntactical correctness (lint + build).
- Check feasibility of Execute functional tests on each report to confirm if Report is working as per expected behaviour.

6.4 Metadata Update (Check if possible):

- Mark each report's "additional settings" as "React-converted" upon successful validation.
- Persist converted code back to the report metadata CLOB.

Note: The Diff Tool team has already built a ChatGPT-4 powered conversion agent (https://difftool-dev.rnd-cloud.metricstream.com/code/js/report-code-assistant). Please review its usage instructions and sample files, and coordinate with . The agent's core logic, scenario handling, and conversion workflows should be used into our Admin-UX Conversion Solution.

Overall Flow



Agent Process Flow - BackBone to React JS conversion

Reference

1. Platform Report APIs - Technical Publications - Confluence

2. Samples are/will be available at Backbone_To_React_Using_report-APIs