**BRE Editor – User Guide (Enhanced for Nontechnical Users)**

*Version: May 2025*

**1. Introduction**

The Business Rule Engine (BRE) Editor is a user-friendly interface to view and edit your organization's policy logic, which is structured in a JSON file. It enables Product, Risk, and Operations teams to make policy-level changes without needing to manually edit raw JSON code.

**2. Installation & First Use**

* Clone the repository:  
  git clone https://github.com/rm1605/brew-ui/tree/version-1
* Navigate to the folder:  
  cd <folder-name>
* Install dependencies:  
  npm install
* Launch the application:  
  npm start
* The UI runs at:  
  <http://localhost:3000>
* The editor is also deployed on Vercel: <https://brew-ui.vercel.app/>

**3. Uploading a Policy File**

* Click **Upload Policy JSON**.
* Select your .json file containing business rules.
* The Rule Table will auto-populate with all the rule entries.

**4. Understanding the Rule Table**

Each rule you see in the UI corresponds to a structured entry in the uploaded JSON under the key ruleUnitDtoList.

**Mapping JSON to UI Elements:**

| **JSON Field** | **Appears in UI As** |
| --- | --- |
| ruleCheckpointParameter | Main field label |
| ruleMetadata.ruleDescription | Tooltip or subtext (where used) |
| ruleConfig.allowedList / blockList | Editable list input with switch |
| operand.value | Hidden logic, not user-editable |
| ruleTemplateGroupCategory | Category filters (e.g. "Deviations", "Bureau Details") |
| ruleId | Internal reference (shown in tooltips or downloads) |

**5. Tabs: Rules vs Deviations**

Use the **Rules** and **Deviations** tabs to switch between standard eligibility rules and policy exceptions. Each section displays only relevant rule types.

**6. Editing a Rule**

* Click any editable field to open the Edit Dialog.
* Editable fields are highlighted in **pale yellow**.
* You can:
  + Enter or change values directly.
  + Toggle between **Allowed** and **Blocked** using the switch.
  + Enter comma-separated values for lists.
* Click **Save** to apply changes.

**7. Filtering for Editable Fields**

Inside the Edit Dialog:

* Use the **"Show only editable fields"** checkbox to focus on what can be changed.
* This filters out read-only metadata or logic for easier editing.
* Collapsible sections (like "operand" or "ruleConfig") will remain open if they contain editable fields.

**8. Deviation Rules in Plain English**

Deviation rules allow for exceptions under specific conditions (like L1 or L2 approvals).

In JSON, these appear as:

"ruleCheckpointParameter": "Drop in Turnover",

"ruleConfig": {

"allowedList": ["L1", "L2"]

}

In the UI, this becomes a rule labeled “Drop in Turnover” with a list input. You can switch between Allowed/Blocked and modify who can approve the exception.

No code or logic editing is needed — just change the allowed levels.

**9. Deleting a Rule**

* Click the trash icon at the row level.
* Confirm in the prompt.
* The rule will be permanently removed from the session (until re-upload).

**10. Attaching Client Requests**

* Upload the corresponding client request (PDF or Excel) via the **Attach Client Request** button.
* The file is linked and included in your download ZIP for traceability.

**11. Exporting Your Changes**

Click **Download Export** and fill in:

* Your name
* Email or initials
* Description of changes

The ZIP will contain:

* old-rules.json – Before your edits
* new-rules.json – After your edits
* diff-summary.xlsx – Changes in Excel format
* audit-metadata.json – Your metadata
* client-request.pdf – Attached request (if any)

**12. Rolling Back**

To revert to a previous version:

* Unzip an old export.
* Re-upload old-rules.json to restore the snapshot.

**13. JSON Glossary (Plain English)**

| **Term** | **Meaning** |
| --- | --- |
| ruleUnitDtoList | List of all rules in the policy |
| ruleCheckpointParameter | Name/label of the condition being evaluated |
| ruleMetadata | Descriptive info like rule name or description |
| ruleConfig | Core of the rule – allowed values, thresholds |
| allowedList / blockList | Who is allowed (or blocked) under this rule |
| operand | Logic evaluator object (not editable) |
| ruleTemplateGroupCategory | Rule grouping/category for filters |
| decision | The system action if rule fails (e.g., REJECT) |

**14. Keyboard Shortcuts**

| **Action** | **Keys** |
| --- | --- |
| Save field | Enter |
| Cancel edit | Esc |
| Expand / Collapse | → / ← |
| Search | Ctrl + F |
| Export | Ctrl + Shift + S |

**15. Best Practices**

* Always attach a client request when editing a rule.
* Export a ZIP file after every batch of edits.
* Keep ruleId unchanged unless absolutely necessary.
* Use the diff summary for approvals and audits.

**16. Appendix – Sample Rule Structure**

{

"ruleCheckpointParameter": "CIBIL Score (Primary Applicant)",

"ruleMetadata": {

"ruleDescription": "Minimum score required for eligibility"

},

"ruleConfig": {

"operator": "GREATER\_THAN\_EQUAL\_TO",

"operatorValue": 750,

"allowedList": [],

"blockList": []

},

"ruleTemplateGroupCategory": "Bureau Details",

"decision": "REJECT"

}

This results in a row titled **CIBIL Score (Primary Applicant)** with a threshold of 750 and editable exception fields.