Rules Engine

[1. Introduction 3](#_Toc468367966)

[2. Rules configuration 3](#_Toc468367967)

[3. Rules validation 4](#_Toc468367968)

[4. Rules dependency checker 5](#_Toc468367969)

[5. Rules processing 6](#_Toc468367970)

# Introduction

The Rules Engine will provide the following features:

* ability to configure many of the custom rules
* ability to configure complex data mappings for the report views
* ability to configure complex validation requirements

# Rules configuration

The Rules can be configured at a Document level (in the Document Design module).

In the first phase, the JSON for each rule will have to be configured by the user, using the user guide and instructions that will be provided. A simple user interface will be provided to add the configured rules to the Document Design.

In a later phase, a user interface can be developed to make the rule configuration easier.

The details of the JSON schema for rules configuration and examples of a few configured rules are provided in the detailed Rules Configuration documentation.

The different elements of the rules configuration are:

|  |  |
| --- | --- |
| Element | Description |
| targetelement | This is the full JSON path of the data element that is the target of this rule. The target element get it’s data/value from the rule, or it may be the subject to a validation rule |
| Triggers | Triggers are the lifecycle events like document opening/saving or section opening/saving that act as triggers for this rule to be executed.  The triggers can also be configured for a set of pre-defined user interface events |
| Actions | Actions are configured to determine what to do after the rule has been executed, and returned an output.  In the case of a rule of type data source, the action may be configured to update the target element with data returned by the rule.  Some predefined actions like select dialog, view dialog can also be configured as actions. |
| Ruleconditions | This element contains the logic for the rule processing. It contains a set of sources to which filter can be applied.  Additional expressions can be configured to merge multiple sources into a singe output.  The output that is returned is processed based on the actions configured for the rule |

# Rules validation

The rule validator is a component that will validate the rule that is being added for:

1. Proper target and source element paths
2. Valid expressions
3. Valid schema
4. Dependency check (see point 4 for details)

The rule can be added only if it is valid for all these conditions.

# Rules dependency checker

The multiple rules that are configured may affect each other in a cyclic way and cause a closed loop.

Explanation:

A rule has a target element and a set of source elements. Two rules ma y be configured where a source in one rule is a target in the other, and vice versa.

This will cause a closed loop of the rules, and they will keep executing in an infinite loop.

Rule A

Target

Source

Rule B

Target

Source

Rule B

**Source A**

Target C

Rule C

Rule A

**Target C**

**Target A**

Target A

Source A

To avoid this scenario, a rules dependency checker will be implemented that will check for such conditions, and will not allow a rule to be added at design time, if it is going to result in a closed loop.

The dependency checker will also sequence the rules to run in the appropriate sequence based on the source and target elements, when there are rules that need to be run on document load (on the GUI) or on backend updates like global updates.

# Rules processing

The rules processing will involve the following steps:

1. Rules Preprocessing/Compilation

Since the rules are common for all documents of a Document Design, the rules will be preprocessed and compiled into a format that will help boost performance of the rules processing.

The rules will also be sequenced, if required, using the dependency checker.

1. Rules Processing

The rules will be executed when the document is rendered in the GUI or is updated using some other process. The rules will be executed based on the triggers that are fired during the document rendering or document update.