muddy boots. online

Muddy Boots Inc.

Rules Engine

V 1.0

Introduction

The purpose of rules is enabling automation within the MuddyBoots platform. At a high level, rules act as black boxes: they take an input and perform actions if their conditions are met. These actions can include generating alerts, creating schedules, assigning work orders, and more. Rules can be triggered by doing various actions in-app (e.g., recording an activity, updating equipment) or by data changes (e.g. imports).

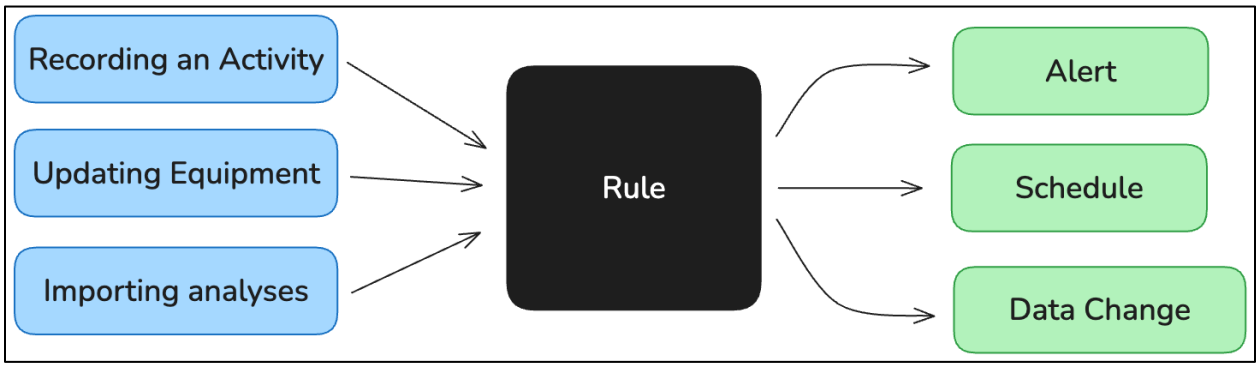


Figure 1. Rule logic flow

Templates

Templates determine the action a rule will perform, and which events will trigger it. Every rule must have a template, and you will be prompted to select one when creating a new rule.

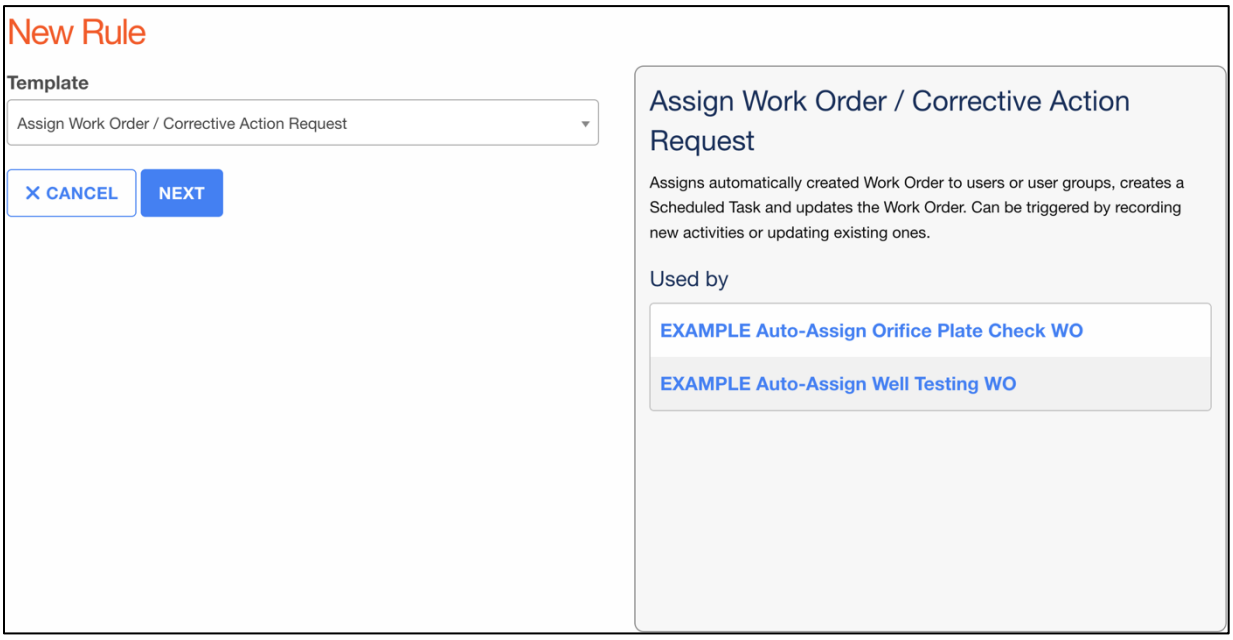


Figure 2. "New Rule" Page: Template Selection

The available templates are defined below, and each template can be triggered by creating or updating activities, unless specified otherwise:

Template Name	Action	Triggered By
Auto-Accept Activity	Accept activities (set the status to "Auto-Accepted")	
SIMOPS Notify	Send an email notification to users from the specified SIMOPS Divisions	
Assign Work Order / Corrective Action Request	Assign Work Orders automatically created from "Needs Attention" values to users and/or user groups. Optionally, update Work Order properties and create a Scheduled Task	
Suppress Work Order / Corrective Action Request	Prevent automatic Work Order creation from "Needs Attention" values.	
Create Scheduled Task	Create a Scheduled Task using the specified configuration. If an active Scheduled Task already exists on the new task's start date, task creation is skipped to prevent duplicates.	
Dissatisfy Schedule	Unlink the triggering Activity from its Scheduled Task, if such a link exists	
Create Custom Alert (Activity) *and* **Create Custom Alert (Equipment)**	Create an alert with the specified title and description	Creating and updating activities or equipment
D017 Schedule Optimization	Generate schedule optimization alerts based on AER Directive 17 Density Exemption Criteria.	

Rule Configuration

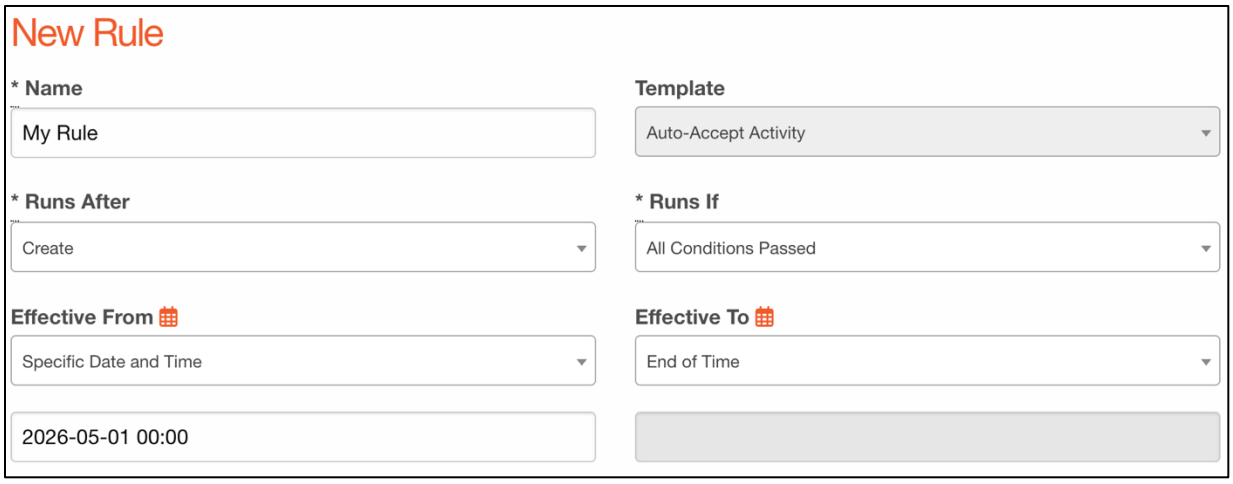


Figure 3. "New Rule" Page.

The following properties comprise the primary rule configuration options:

Name	The name of the rule. It will be shared across all versions of the rule.
Template	Determines what actions the rule will perform and how it can be triggered.
Runs After	Determines the actions after which the rule will run.
Runs If	Determines if the rule needs all conditions to pass to run or at least one.
Effective From	Date and time when the rule becomes effective
Effective To	Date and time when the rule stops being effective

Scoping

Scopes control what records can trigger rules. For example, if a rule is scoped to an Activity Type, only records for that Activity Type will trigger it; all others are ignored.

Figure 4. Scopes page for "New Rule".

Scopes are always evaluated altogether. If a rule is scoped by Activity Type and Field, the trigger must match both scopes for the rule to run.

The available scopes can be different from rule to rule and are determined by the selected template. If a scope is left blank (i.e. nothing is selected), it is seen as inactive. Inactive scopes are skipped and do not have any effect on the rule.

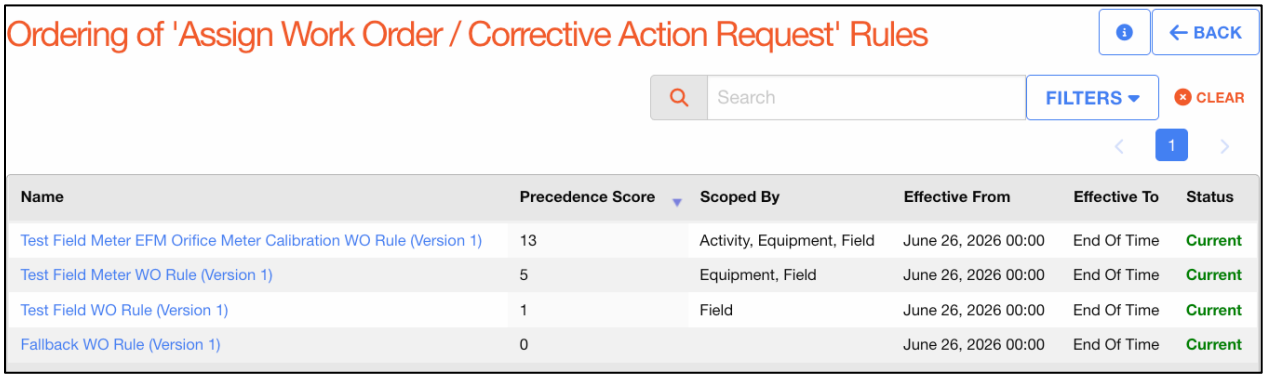
Scoping Precedence and Precedence Score

Scoping Precedence defines which scopes take priority over others. It ensures that more specific (tightly scoped) rules execute instead of broader ones, if scopes permit so.

Precedence Score is an integer that determines each rule's priority. Higher scores indicate higher precedence.

The following example, shown in Figure 5, shows several rules with different scopes. The template's precedence order is:

Activity Question → Activity (Activity Type) → Equipment → Site → Field



Name	Precedence Score	Scoped By	Effective From	Effective To	Status
Test Field Meter EFM Orifice Meter Calibration WO Rule (Version 1)	13	Activity, Equipment, Field	June 26, 2026 00:00	End Of Time	Current
Test Field Meter WO Rule (Version 1)	5	Equipment, Field	June 26, 2026 00:00	End Of Time	Current
Test Field WO Rule (Version 1)	1	Field	June 26, 2026 00:00	End Of Time	Current
Fallback WO Rule (Version 1)	0		June 26, 2026 00:00	End Of Time	Current

Figure 5. "Assign Work Order / Corrective Action Request" Precedence

If a triggering action is performed (e.g. a new activity is recorded), the Rules Engine will look up what rules it needs to run. One of the steps in this process involves evaluating scopes. Here, if the scopes of the first rule ("Test Field Meter EFM Orifice Meter Calibration WO Rule") are met, the engine will proceed only with this rule and disregard all other due to them having a lower Precedence Score. If the scopes are not met, it will continue evaluating rules in the order of descending Precedence Score.

Effective Periods

An Effective Period defines the time range during which a rule version is active. Effective Periods use semi-open intervals, meaning the end datetime is excluded.

For example, a period from Jan 1, 2026, 00:00 to Feb 1, 2026, 00:00 is effective from Jan 1, 2026, 00:00 to Jan 31, 2026, 23:59. (Note: Every version must have an effective period of at least 1 minute.)

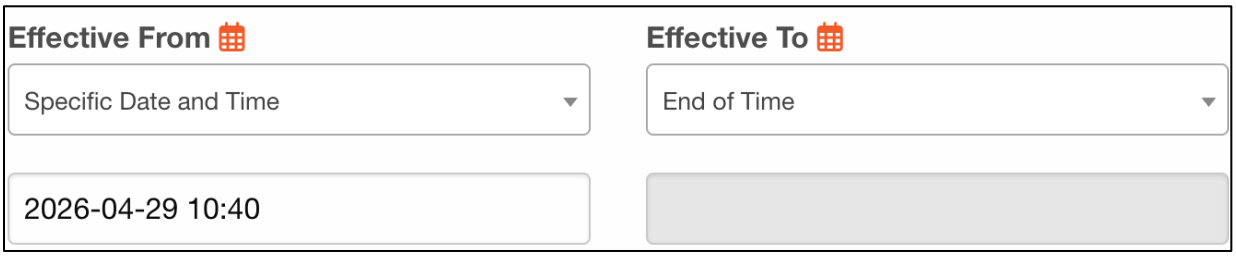


Figure 6. Effective Period Inputs, "New Rule" page.

Effective Periods follow these constraints:

1. Versions cannot overlap – only one version can be effective at a time.
2. All runs of a specific rule must fall within its effective period.
3. The effective period must contain the required period, if one exists.

The **required period** is determined by runs. Runs must always be within the effective period of their rules and this shapes the required effective period. It stretches between the first and the last chronological run and must be fully encompassed by the effective period of its rule.

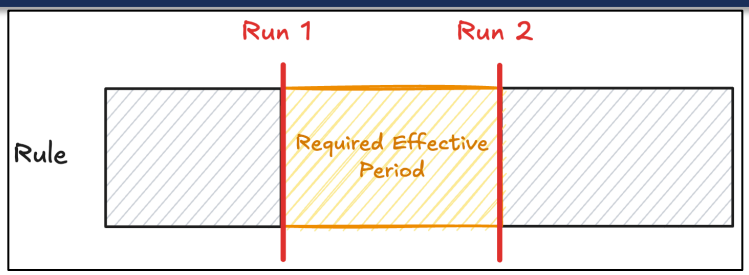


Figure 7. Required Effective Period based on Runs

While overlaps are not allowed, they can be created temporarily. The system will automatically resolve them when possible, however, only partial overlaps can be resolved automatically. If a version fully overlaps another version or it is fully overlapped by one, effective periods must be adjusted manually.

Partial overlap resolution will fail if adjusting sibling versions conflicts with their runs.

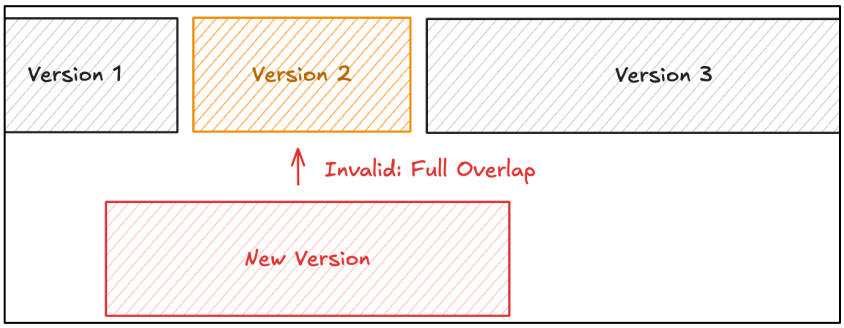


Figure 8. Invalid Version: Fully Overlaps another version

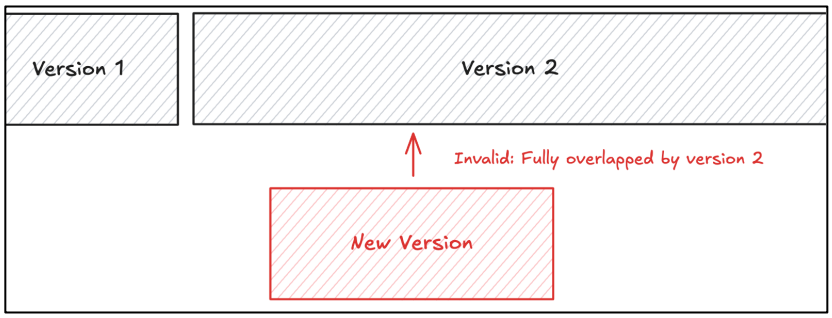


Figure 9. Invalid Version: Fully Overlapped

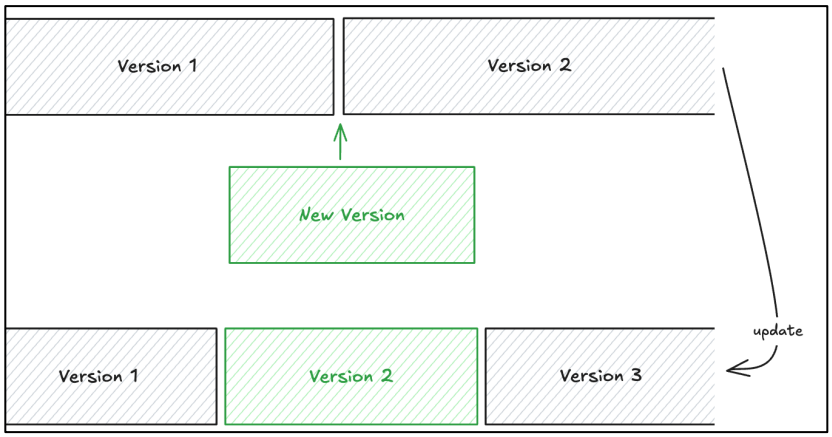


Figure 10. Valid New Version

Versioning

Rules perform automated data changes and, to ensure historical data persists, rules are versioned. Additionally, versioning helps with running correct rules on historical activities where they will trigger the rules that were active at the time of the *Activity Date*.

If a rule has at least one run, it will be seen as historical. Such rules cannot be updated and will require a new version to be created if its properties need to be changed.

You can create future versions if you know that your requirements will change at a specific date. When doing so, the future version will become active once the "Effective From" date and time is reached. Versioning is not the same as cloning. Creating a new version creates it within the same rule while cloning creates a totally separate rule.

Conditions

Conditions are sections of logic that can be used to specify under what conditions a rule needs to run. A condition can be defined that ensures a rule only runs if a form element value matches a certain criterion. Conditions always evaluate to a Boolean value. In other words, rules require conditions to be true in order to run.

Conditions are made of graphs that contain nodes (Comparison, Function + Data Source, Arithmetic, etc.). Every node has input(s) that it uses to perform actions and produce an output that can be passed over to a different node.

In the example below, "Function + Data Source" node is grabbing the value of "All Volumes Within 10% of Previous Test" and passing it to the "Comparison" node where it gets compared with "Fail" and the result of this comparison is passed to "End", becoming the result of the condition. This condition will evaluate to *true* if the value of the specified form element is "Fail" and *false* if the value is not "Fail" (e.g. "Pass").

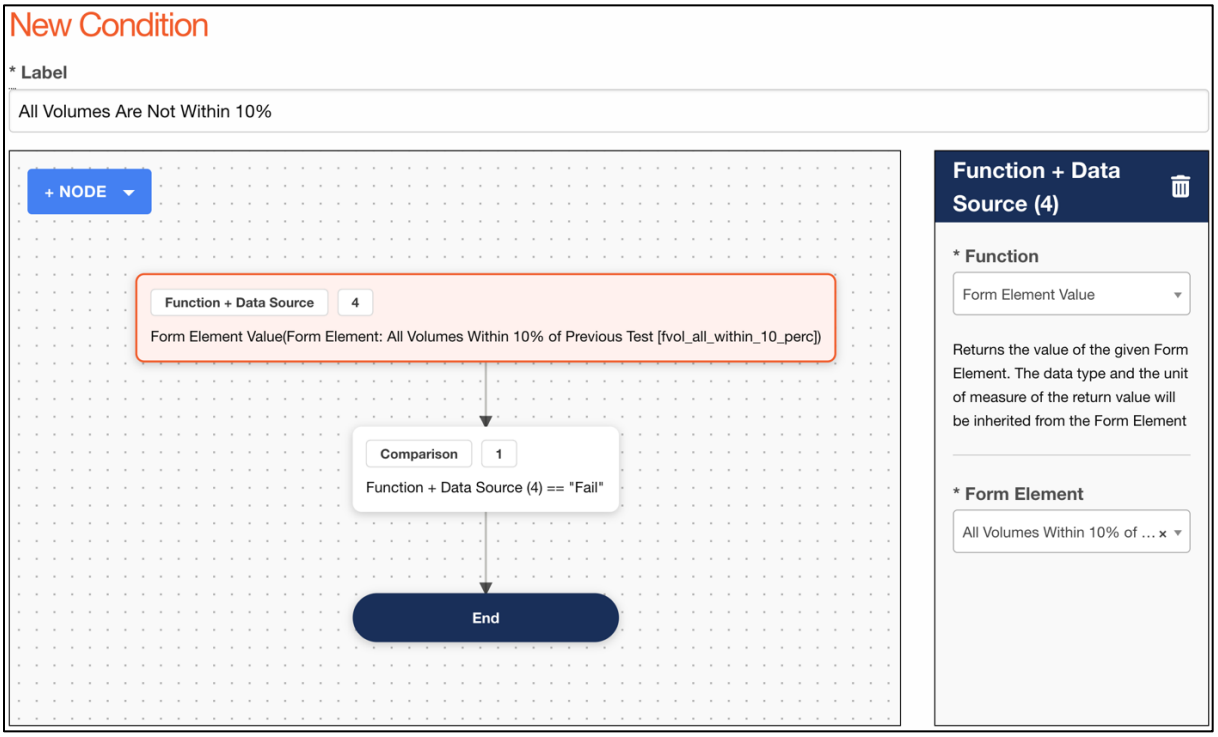


Figure 11. "New Condition" Page: Condition Builder

Please Note: It is recommended not to make a condition too big and break them up into smaller ones, when possible.

Supported Nodes

Arithmetic

Performs arithmetic operations (addition, subtraction, multiplication, etc.).

When performing operations on numbers, be mindful of units. For addition and subtraction, both operands should either have units or be unitless. If both operands have units, the units must either be the same or be convertible to each other.

- 34 cm + 1 m -> **134 cm**
- 1 psi + 1 -> **Error**

Multiplication, and division operations require the right operand to be a unitless number. The system does not support instances that convert the type of unit. For example, it does not support converting length (cm) to an area (cm^2).

- 2 m * 5 -> **10 m**
- 50 cm * 2 cm -> **Error**

In addition to numeric operations, you can use addition and multiplication with strings. Addition concatenates two strings, while multiplication repeats a string '*n*' times.

- "hello" + "world" -> **"helloworld"**
- "test" * 3 -> **"testtesttest"**

Comparison

Performs comparison (less than, greater than, etc.) and equality (equals, not equals) operations.

All available operations are valid for numbers. However, comparing a unitless number with a number that has a unit is not allowed. When performing operations on numbers, both operands should either be unitless or have units.

When comparing two numbers with units, their units must be convertible to each other.

- 1 m == 100 cm -> **True**
- 1 $CAD == 50 km/h -> **Error**
- 1 $CAD == 10 -> **Error**

Comparison and equality operations can also be performed on strings. String comparisons are based on alphabetical (lexicographical) order. Comparing a string with a non-string value will result in an error.

- "abc" < "bcd" -> **True**
- "1" == 1 -> **Error**

Lastly, this node can be used to perform equality operations on Booleans. Equality operations are allowed, but comparison operations on Booleans will result in an error.

- false == false -> **True**
- false < true -> **Error**

Logical

Performs logical "AND" and "OR" operations.

This node accepts only Boolean values. Using a value of any other type (e.g. a non-Boolean value passed from a Function + Data Source node) will result in an error.

- true AND false -> **False**
- true OR false -> **True**
- 1 AND true -> **Error**

Inclusion

Performs inclusion and exclusion checks on strings by determining whether one string is contained within another. This node accepts string values only. Using a value of any other type (e.g. a non-string value passed from a Function + Data Source node) will result in an error.

- "compressor" ~ "press" -> **True**
- "meter" ~ "orifice" -> **False**
- 123 ~ "23" -> **Error**

Function + Data Source

Runs a variety of functions, ranging from functions that modify the input value (e.g. Square Root) to functions that retrieve values from Activities, Equipment, etc. (e.g. Form Element Value).

The inputs and output of this node depend on the selected function. For more information about a function, see its description, which appears when the function is selected. The list of available functions depends on the Template used by your Rule.

Examples:

- sqrt(81) -> **9**
- percent_change(5 cm, 7.5cm) -> **50 %**
- form_element_value(Form Element: Volume [meter_vol]) -> **15 m^3** *(If the value in the activity is 15 m^3)*

Operation Symbols

There is support for the following math operations using the specified symbols, unless specified otherwise:

Operation	Symbol	Applicable Nodes
Addition	+	Arithmetic
Subtraction	-	Arithmetic
Multiplication	*	Arithmetic
Division	/	Arithmetic
Greater Than	>	Comparison
Greater Than or Equal To	>=	Comparison
Less Than	<	Comparison
Less Than or Equal To	<=	Comparison
Equal To	==	Comparison
Not Equal To	!=	Comparison
Includes	~	Inclusion
Excludes	!~	Inclusion
And	AND	Logical
Or	OR	Logical

Available Functions

The following functions are supported by *Function + Data Source* nodes. They are available to rules triggered by activities, unless specified otherwise:

Name	Description	Available to
Absolute Value	Calculates the absolute value of the given number. Returns a number with the unit taken from the input value.	
Square Root	Calculates the square root of the given number. Returns a number with the unit taken from the input value.	
Absolute Difference	Calculates the absolute difference of two given numbers. Returns a number with the unit taken from the left operand.	
Percent Difference	Calculates percent difference of the two given numbers. Returns a number with unit of measure set to '%'.	
Percent Change	Calculates the percent change of the two given numbers. Returns a number with the unit of measure set to '%'. The function will fail if the old value is 0 but the new value is not due to zero division.	
Form Element Value	Returns the value of the given Form Element. The data type and the unit of measure of the return value will be inherited from the Form Element.	
Form Element Past Value	Returns the value of the given Form Element from the previous activity. The data type and the unit of measure of the return value will be inherited from the Form Element.	
Standard Deviation	Calculates the standard deviation of Form Element values from previous accepted activities. The data type and the unit of measure of the return value will be inherited from the Form Element.	Auto-Accept Activity
Equipment Property Value	Returns the value of the given Equipment Property. The data type and the unit of measure of the return value will be inherited from the Equipment Property.	Create Custom Alert (Equipment)
Equipment Property Changed?	Checks if the last update to the piece of Equipment changed the value of the specified Equipment Property. Returns a boolean.	Create Custom Alert (Equipment)
Trigger Value	Returns the value of the Form Element that triggered the rule ('Needs Attention' and its variants). The data type and the unit of measure of the return value will be inherited from the Form Element.	Assign Work Order / Corrective Action Request, Suppress Work Order / Corrective Action Request

Log Statuses – Conditions

"**Pass**" is shown a condition evaluates as *true*.

Condition	Status	Message
Test Results == Fail	**Pass**	*N/A*

"**Fail**" is shown when a condition evaluates as *false*.

Condition	Status	Message
Test Results == Fail	**Fail**	*N/A*

"**Error**" is shown when a condition cannot be evaluated due to a configuration error or lack of data. While it is acceptable to see this status, it usually indicates that the condition needs to be updated.

Condition	Status	Message
Test Results == Fail	**Error**	Comparison (1) - Cannot perform the given operation on String ("Fail") and Number (1234.0)

Log Statuses – Rules

Success

"Success" is shown when a rule meets its conditions and successfully performs its action (e.g. creates a Scheduled Task).

Run Date	Result Date	Rule	Template	Result
July 21, 2026 14:52	July 21, 2026 14:52	EXAMPLE (Version 2)	Create Scheduled Task	**Success**

No Changes

"No Changes" is shown when a rule meets its conditions but does not make any changes to the data.

Example: A "Create Scheduled Task" rule attempts to create a Scheduled Task for a date on which an active Scheduled Task already exists. Since the rule does not create a duplicate task, "No Changes" is shown.

Run Date	Result Date	Rule	Template	Result
July 21, 2026 15:03	July 21, 2026 15:03	EXAMPLE (Version 2)	Create Scheduled Task	**No Changes**

Conditions Failed

"Conditions Failed" is shown when a rule does not meet its conditions. This may occur because a condition evaluates to false or because a condition cannot be evaluated due to a configuration error.

You can find more details about which conditions passed and which did not, as well as any error messages for conditions that could not be evaluated, in the "Condition Results" table.

Examples of when a "Conditions Failed" log would be created:

- A condition expects the value of "Total Tank Volume" to be greater than 100 m^3, but the actual value is 90 m^3.
- A condition compares values that cannot be compared (e.g. a unitless number and a number with a unit).

Run Date	Result Date	Rule	Template	Result
July 21, 2026 14:35	July 21, 2026 14:34	EXAMPLE (Version 1)	Create Scheduled Task	Conditions Failed

Warning

"Warning" is shown when a rule meets its conditions but cannot perform its action due to a configuration issue. This usually indicates that the rule needs to be updated.

Example: "Warning" is shown if a "Create Scheduled Task" rule does not have a Site specified. This could occur if the Site referenced by the rule has been deleted.

Run Date	Result Date	Rule	Template	Result
July 21, 2026 15:19	July 21, 2026 15:19	EXAMPLE (Version 1)	Create Scheduled Task	Warning

Error

"Error" is shown when an unexpected server error occurs. If you see an "Error" log, please contact Muddy Boots Support.

Do not confuse this with the "Error" status for conditions. Unlike a rule-level "Error" log, it is acceptable for an individual condition to have an "Error" status.

Run Date	Result Date	Rule	Template	Result
July 21, 2026 15:21	July 21, 2026 15:21	EXAMPLE (Version 1)	Create Scheduled Task	Error

Run Log

Every rule run is written to the log, making it easier to track down what rules are doing and whether they are functioning as intended.

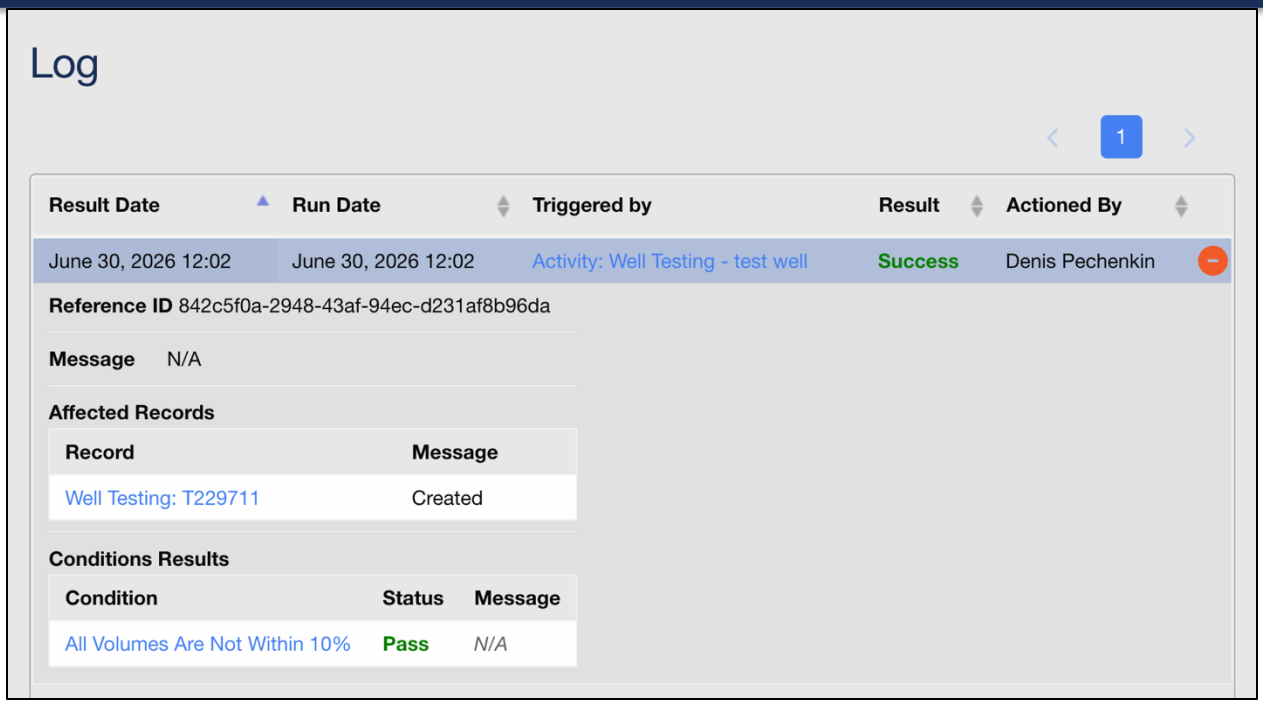


Figure 12. "View Rule" page: Run Log

Every run contains information about what triggered it, the status and the error message (if applicable). If a rule has conditions, their statuses will also be shown here. If a condition cannot be evaluated due to a configuration error, the error details be shown here as well.

Additionally, runs include information about what data was changed and what records were affected. This can be found under the "Affected Records" table.