Expressions

10/11/2026 2:58 pm GMT

Expressions are predominantly used to support routing decisions in a workflow or to control what actions are displayed and when you're allowed to view, start or finish the action.

Expressions let you build true / false conditions that drive decisions in a workflow. Each expression evaluates to either **TRUE** or **FALSE**, and the workflow uses that result to decide what happens next as a product moves through manufacturing.

An expression can combine logic from more than one piece of data, reference a SQL statement, or check the skills of the current user.

You can apply an expression in three ways:

1. Start and completion control

Use an expression to control whether an operation, or an action within an operation, can be started or completed.

Example: An operation can only be started a minimum of 30 minutes after the previous operation was completed.

2. Skip logic

Use an expression to create a dynamic sequence of actions that depends on the data gathered up to that point.

Example: The operation is "Bake cake":

- Action 1: Always display instructions to load into oven and set correct program on oven (no start expression).
- Action 2: Record unload from oven.
- Action 3: If bake time is out of tolerance, prompt for a reason and comment, and record an appearance quality check. The data recorded can be used for a conditional flow.
- Action 4: If the reason is overbaked in error, route to scrap. If overbake was intentional, route to the next step.

3. Conditional flows

Use an expression to decide the next step in a routing.

Example: Go to rework if any check in the completed set has failed.

How an expression is structured

An expression has two parts:

- **Expression header**: The name and settings of the expression. This is what you select when you reference the expression in a workflow.
- **Expression elements**: The logic that decides whether the expression is TRUE or FALSE. An element's **Field Type** sets the kind of logic it uses:
  - **Case Data**: Tests a case data value on the item.
  - **SQL Statement**: Runs a SQL statement against live data.
  - **Current User Skill**: Checks the skills of the user working on the job.

An expression can contain more than one element.

You can combine the logic from more than one item to create a compound rule using **And / Or** to join them.

Create an expression header

Create the header first. The header is a top-line record in the top grid of the Expressions screen.

1. Navigate to the **Expressions** screen, add a new expression in the top grid. The **Edit Expression** dialog opens.

2. Complete the fields described in the table below.

Add / Edit Expression

Blue fields in the form are required and are flagged with an asterisk (*) in this document.

- **SUID***: The identifier of the expression.
- **Name***: Make sure this is intuitive and self explanatory. When you reference this in the workflow the expression name is the only information you will have to go by, e.g. in the example here, the name is 'is >30mins from previous op', i.e. true would be >30 mins, false would be <30 mins. If you called this 'Duration from previous op' it would be unclear what the criteria for true / false was.
- **Description***: A description of the expression and its true/false criteria.
- **Failure Message**: This is only needed where the expression is being used to control what the operator does and you want to communicate failure information to them. For example: "You are not allowed to book onto op 20 if insufficient time (15 minutes) elapsed since completing op 10".
- **Controlled***: Only controlled expressions are able to be used in a controlled workflow. If an expression is controlled then additional granular permissions are required to make changes.

Check where an expression is used

Before you change an expression, select **Where Used** on the **Expressions** screen to see the impact of the change. SQL statements within expressions can also be controlled.

Add logic to an expression

When the header exists, add one or more elements to it. Each element is defined in the **Edit Expression Element** dialog and appears in the **Elements** grid for the expression.

1. Select an expression header and click the **New** button in the **Elements** section of the screen.

Add / Edit Expression Element

Blue fields in the form are required and are flagged with an asterisk (*) in this document.

- **Field Type***: The kind of logic: Case Data, SQL Statement or Current User Skill.
- **Field Value***: The item being assessed. For case data, this is the case data field. For a SQL statement, this is the SQL statement.
- **Field Data Type***: The type of data you expect, for example Text or Boolean.
- **Operator***: The comparison to apply, for example *contains* for case data or *returns record* for a SQL statement.
- **Condition Value**: The value to compare against.
- **And/Or**: How this element combines with the next element in a compound rule.
- **Case Sensitive** checkbox: Select to make the comparison case sensitive.
- **Left Bracket / Right Bracket** checkboxes: Select to group elements when you build compound rules.

Option 1: Case Data

Case data is the simplest type of expression logic. You define which data you are assessing, the type of data you expect, and the logic to apply to it. For more information, see Case Data.

1. In the **Edit Expression Element** dialog, set **Field Type** to **Case Data**.

2. Set **Field Value** to the case data field to assess.

3. Set **Field Data Type** to the type of data you expect (Boolean, numeric, text, date).

4. Set **Operator** and enter the **Condition Value**.

5. Select **Save**.

Example: The expression is TRUE when the item's *Chair inspection fail decision* case data value contains `REWORK`. To build this, set the Operator to *contains* and the Condition Value to `REWORK`.

Option 2: SQL Statement

Use a SQL statement when the logic is not linked to case data.

1. In the **Edit Expression Element** dialog, set **Field Type** to **SQL Statement**.
2. Select the SQL statement as the **Field Value**, or define a new one in the **Edit SQL** dialog via the **SQL Statement** screen.
3. Set **Field Data Type** field to **Boolean** and **Operator** field to **returns record**.
4. Select **Save**.

How SQL expressions evaluate

- The statement takes live data from the item and returns a Boolean result.
- If the statement **returns a record**, the result is **TRUE**.
- If the statement **returns no record**, the result is **FALSE**.

Link the SQL to the item

To tie the statement to the item, or to the position of the item in the workflow, use the workflow node visit ID parameter `:WORKFLOWNODEVISITID` (see the parameter reference below).

Simplified pattern:

```
select *
from workflownodevisit
where workflownodevisit.id = :WORKFLOWNODEVISITID
```

This pattern is for illustration only. Add the joins and conditions that your own logic needs. The statement must return a record when the condition is met and no record when it is not.

Note: Consult Eyelit Support before you work with SQL statements in expressions.

SQL parameter reference

The parameters available to a SQL statement depend on the function to which the expression is applied. Check this table before you write the statement.

Function	WORKFLOWNODEVISITID	MATERIALITEMID	WORKFLOWTASKID	ACTIONIDSOURCE	WORKFLOWNODEVISITIDX_LOGINID	X_USERID
Action Complete	✔ Yes	✔ Yes	✔ Yes	✔ Yes	✘ No	✔ Yes	✔ Yes
Action Start	✔ Yes	✔ Yes	✔ Yes	✔ Yes	✘ No	✔ Yes	✔ Yes
Action Skip	✔ Yes	✔ Yes	✔ Yes	✔ Yes	✘ No	✔ Yes	✔ Yes
Flow	✘ No	✔ Yes	✔ Yes	✘ No	✔ Yes	✔ Yes	✔ Yes
Operation Complete	✔ Yes	✔ Yes	✔ Yes	✘ No	✘ No	✔ Yes	✔ Yes
Operation Start	✔ Yes	✔ Yes	✔ Yes	✘ No	✘ No	✔ Yes	✔ Yes

Option 3: Current User Skill

Use Current User Skill to create detailed controls logic within workflow actions, e.g. where you have specific skills you want to link to individual actions in a workflow operation. You could create this expression and use it as a start condition on the specific action.

1. Create the expression header. Use a name that states the skill, for example *Senior Machining Qualified*.
2. Add an element with **Field Type** set to **Current User Skill**.
3. Select the skill to check, and add further elements for further skills.

This is useful for detailed control logic within workflow actions. For example, if specific skills are linked to individual actions in a workflow operation, you can:

- Use the expression as a **start condition** on the specific action.
- Use it to decide **skip** logic.
- Use it to decide **conditional flows**.

Tip: Add a **Failure Message** to a skill expression so that operators who lack the skill see why they cannot continue.

Note: You can also assign and manage skills control at workflow node level without using expressions.

Tips

- **Name for the outcome**: Write the name so TRUE is clear, for example *is >30mins from previous op*.
- **Check impact before you edit**: Use **Where Used** to see which workflows depend on an expression.
- **Use only parameters your function supports**: A parameter marked ***✗ No*** in the reference table is not available for that function.
- **Get SQL reviewed**: Consult an experienced implementation consultant or 2nd level support.