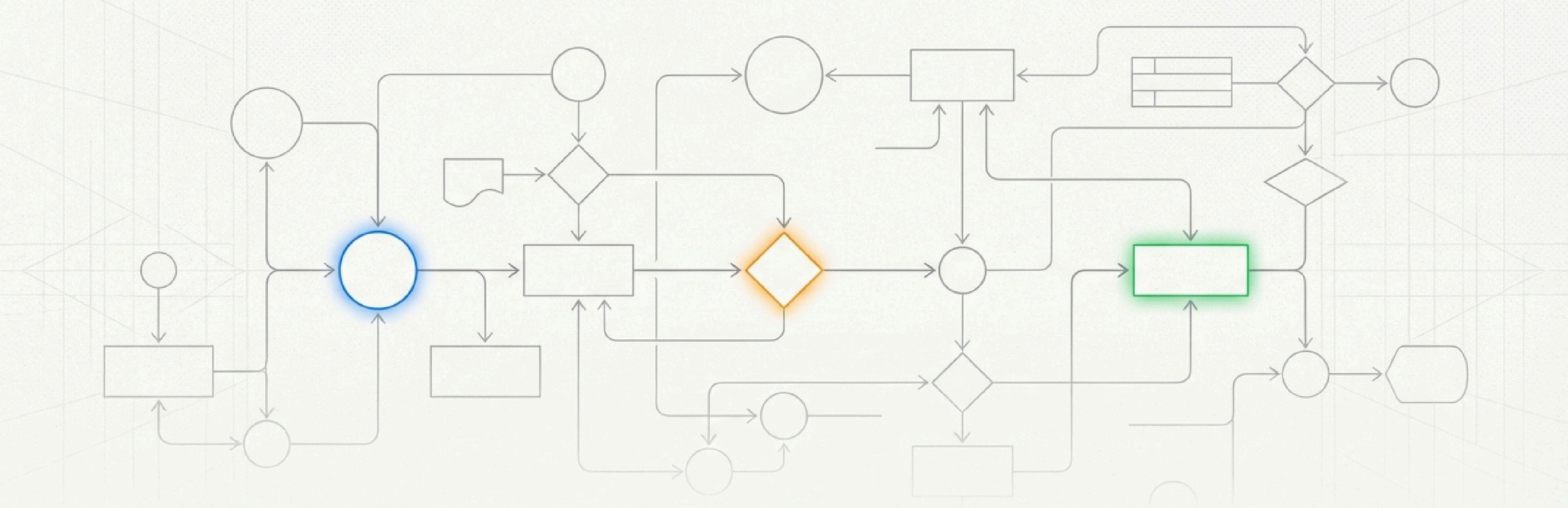
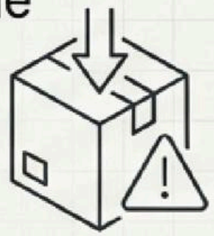


Automate Your Workflow: A Strategic Guide to User Defined Actions in Ostendo

Transform manual processes into intelligent, automated rules without writing a single line of external code.

What if your system could react to your business in real-time?

User Defined Actions allow you to build custom logic directly into Ostendo. You define the trigger and the action, creating a system that automatically responds to events as they happen

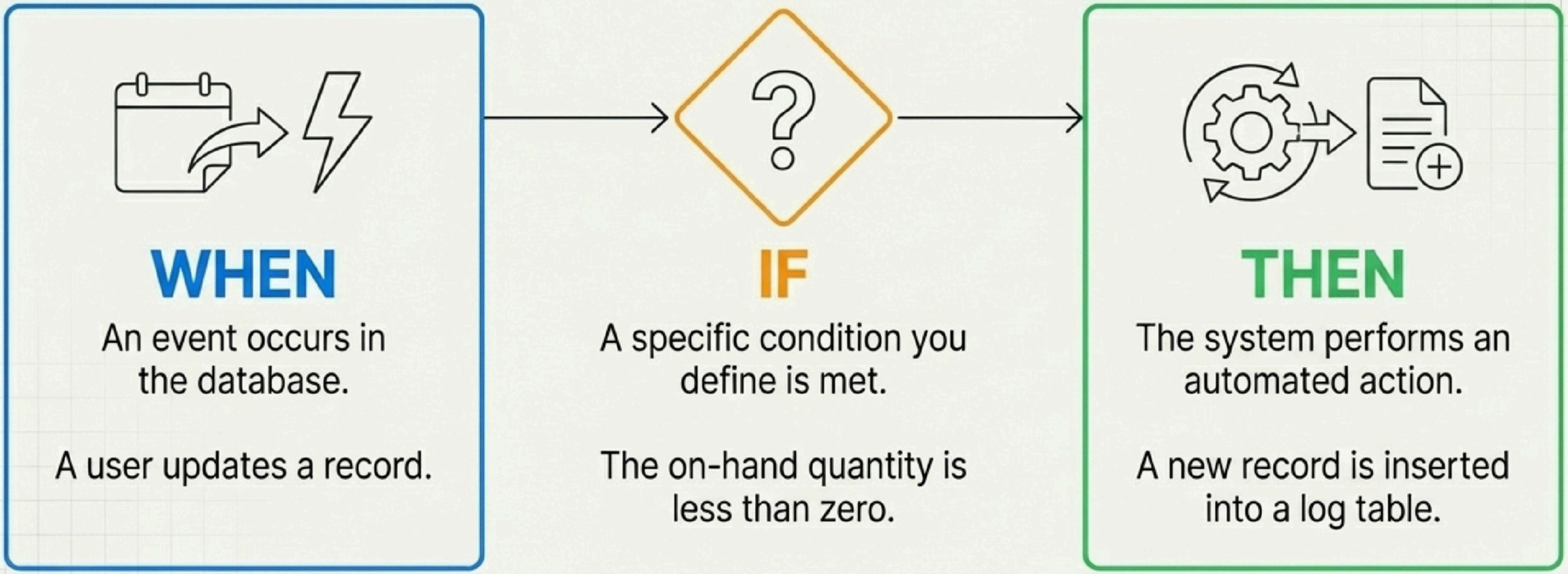


- **Automatically create** a follow-up Call Ticket **when** a new customer is added.
- **Log** every instance of an item's **stock falling below zero**.

- **Create** a history note for **specific critical events**.
- **Initiate** a custom stored procedure **based on a data change**.

Move from reactive data entry to proactive, automated workflow.

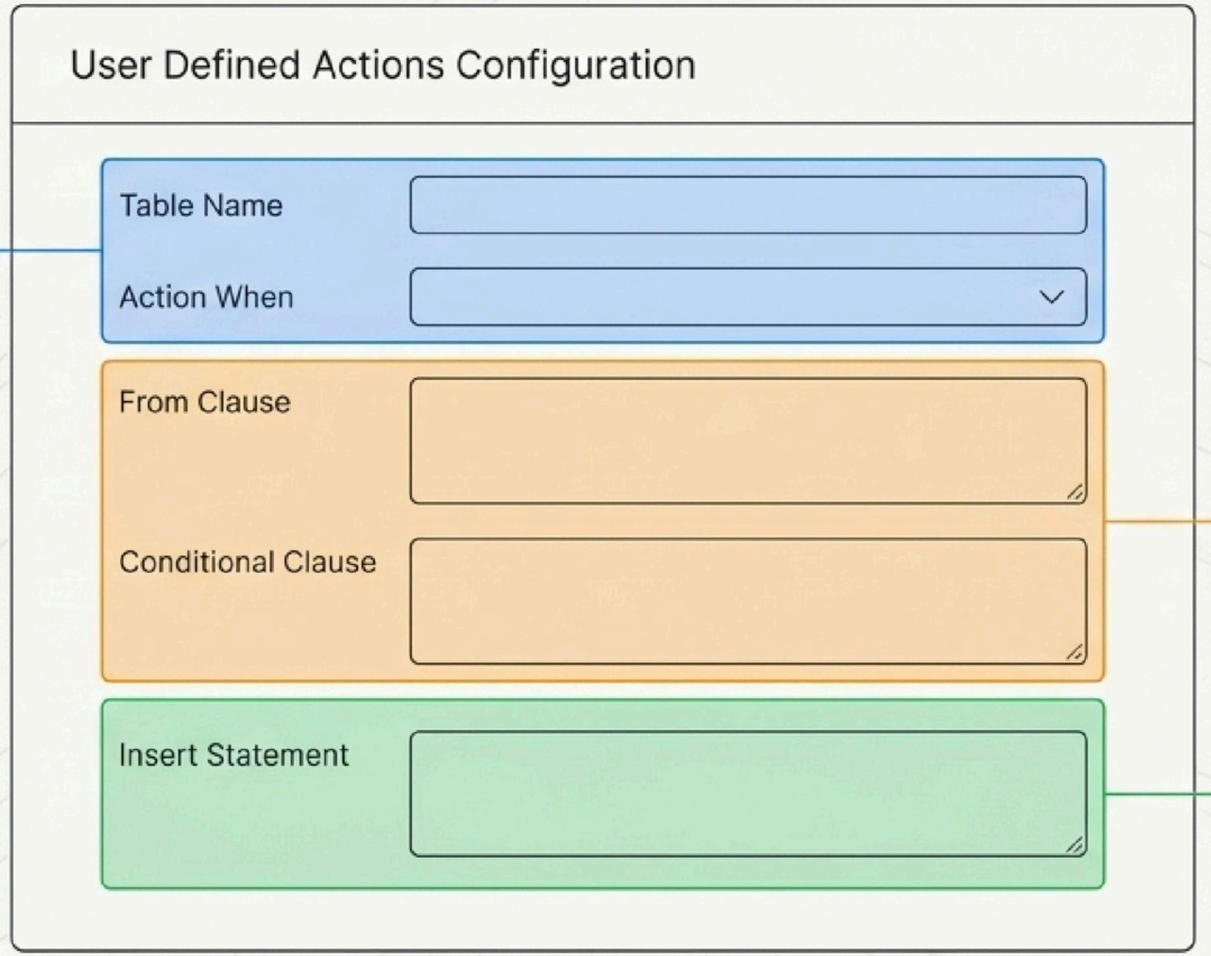
The Core Logic: Every Action is a Simple Three-Part Rule



We will use this 'WHEN, IF, THEN' model to build and understand every User Defined Action.

The Anatomy of a User Defined Action

This is your **WHEN**. You define which table to monitor and what event to listen for (Insert, Update, or both).



This is your **IF**. You specify the logical condition that must be true for the action to trigger.

This is your **THEN**. You define the exact `NSERT` statement the system will execute.

Step 1: Defining Your Trigger (The WHEN)

The first step is to tell Ostendo which table to watch and what specific database event should start the evaluation process.

Table Name	Action When
Select the primary table where the triggering event will occur. This is the focus of your rule. Example: `ITEMMASTER`, `CUSTOMERMASTER`	Specify the precise event that activates the rule. • **'On Insert and Update' (Default)**: The rule runs when a new record is created AND when an existing record is changed. • **'On Insert Only**: The rule runs ONLY when a new record is created. • **'On Update Only'**: The rule runs ONLY when an existing record is changed.

Step 2: Setting Your Condition (The IF)

The conditional clause is the “brain” of your action. It’s a standard SQL `WHERE` clause. The action will only run if this condition evaluates to true. If this is left blank, the action will run every time the “WHEN” event occurs.

`From Clause`

Specify the table name(s) used in your conditional clause.

```
`ITEMMASTER`
```

`Conditional Clause`

The conditioning syntax from a SQL `WHERE` clause.

```
`ITEMMASTER.ONHANDQTY < 0`
```

Multi-Table Syntax

When your `From Clause` or `Conditional Clause` references more than one table, you **must** prefix each field name with its corresponding `TableName` (e.g., `JobHeader.SalesPerson`). This prevents ambiguity and ensures your rule functions correctly.

Step 3: Creating the Outcome (The **THEN**)

This is where you define what happens when your condition is met. The system will execute this SQL `INSERT` statement precisely as written.

Detailed Field Breakdown

- **`Insert Statement`**
  Defines the `INSERT` SQL statement to be executed.

Syntax Guide

For Dynamic Field Values: Always enclose field names in square brackets `[]`.
`...VALUES ([ITEMMASTER.ITEMCODE], [ITEMMASTER.ONHANDQTY])`

For Static/Literal Values: Enclose text or date literals in single quotes `''`.
`...VALUES ('Phone', 'now', 'Open', 'Customer')`

```
INSERT INTO CALLNOTES (CALLSTATUS, COMPANYNAME, CALLBRIEFDESCRIPTION)
VALUES ('Open', [CUSTOMERMASTER.CUSTOMER], 'Send out Information Pack')
```

Automation Blueprint #1: Log Negative Stock Quantities

Scenario

We need to automatically create a record in a custom table (OSTDEF_NEGQTYS) every time an item's on-hand quantity is updated to a value less than zero.

Deconstruction

- **WHEN** (The ITEMMASTER table is updated (On Update Only))
- **IF** (The on-hand quantity is less than zero (ITEMMASTER.ONHANDQTY < 0))
- **THEN** (Insert the item code and quantity into our log table.)

Implementation Table

	FIELD	VALUE
	Table Name	`ITEMMASTER`
	Action When	`On Update Only`
	From Clause	`ITEMMASTER`
</>	Conditional Clause	`ITEMMASTER.ONHANDQTY < 0`
	Insert Statement	`INSERT INTO OSTDEF_NEGQTYS (ITEMCODE, ONHANDQTY) VALUES ([ITEMMASTER.ITEMCODE], [ITEMMASTER.ONHANDQTY])`

Automation Blueprint #2: Create a New Customer Welcome Ticket

Scenario

We want to automatically create an open "Call Ticket" to "Send out Information Pack" for every single new customer added to the system.

Deconstruction

- **WHEN** A new record is added to the `CUSTOMERMASTER` table (`On Insert Only').
- **IF** (No condition). This action should apply to *every* new customer.
- **THEN** Insert a new record into the `CALLNOTES` table with predefined details.

Implementation Table

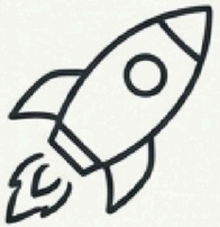
FIELD	VALUE
Table Name	`CUSTOMERMASTER`
Action When	`On Insert Only`
From Clause	`CUSTOMERMASTER`
Conditional Clause	*(Leave Blank)*
Insert Statement	`INSERT INTO CALLNOTES (CALLMETHOD, CALLDATE, CALLTIME, CALLSTATUS, COMPANYTYPE, COMPANYNAME, CALLBRIEFDESCRIPTION) VALUES ('Phone','now','now', ,'Open','Customer',[CUSTOMERMASTER.CUSTOMER],'Send out Information Pack')`

A Professional's Workflow: Build, Test, and Deploy with Confidence

Never test new logic on all users at once. Use the built-in status controls to manage the lifecycle of your action rules safely.

1. DEVELOP & TEST

Set `Rule Status` to **`Restricted to User`**. Specify your own username in the `Restricted User` field. The rule is now active ONLY for you. You can test and refine it without impacting anyone else.



2. DEPLOY

Set `Rule Status` to **`Active For All Users`**. The rule is now "live" and will trigger for all users in the database.

3. DISABLE

Set `Rule Status` to **`Not Active`**. The rule is temporarily turned off. It remains in the system for future use but will never be triggered.

Pro-Tips for Robust Rule Design

Tip 1: Test Your Logic First

- **The Why**
  Before committing to an Action, ensure your `From` and `Conditional` clauses work as expected.
- **The How**
  Use the **Data Spreadsheet (`General -> Data Spreadsheet`)** feature. Write your logic as a `SELECT` statement to preview the results and debug syntax errors in a safe environment.

Tip 2: Defensively Handle Blank or Null Values

- **The Why**
  A field might be empty (`''`) or a true `NULL`. A standard equality check (`= ''`) can fail on `NULL` values.
- **The How**
  Use the `coalesce` function. It provides a default value if the field is null, making your condition reliable.
- **Syntax Example**

```
coalesce(CUSTOMERMASTER.CUSTOMEREMAIL, '') = ''
```

DEFAULT

ⓘ **Note**
`coalesce` is not needed for Lookup or Domain value fields.

Unlocking Advanced Automation with Stored Procedures

For actions that require complex logic, multiple steps, or operations other than `INSERT`, you can call a custom stored procedure. This provides virtually unlimited power.

The 'For Technical Users Only' Method

- Instead of `INSERT INTO...`, your statement will use the `EXECUTE PROCEDURE` command.

Syntax Breakdown

- **Command:** `EXECUTE PROCEDURE PROCEDURENAME`
- **Parameters:** Follow the procedure name with a comma-separated list of its required parameters. Use `[TableName.FieldName]` for dynamic values and `literals'` for static ones.

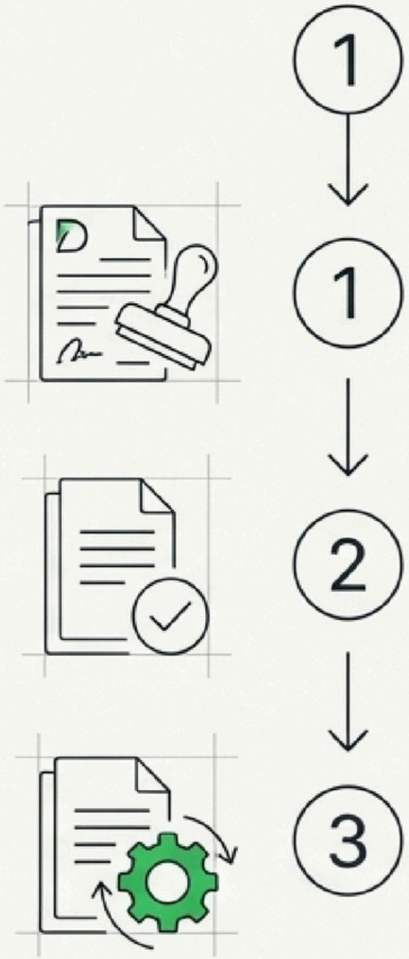
Example from Source

Calling a procedure named `PROCEDURENAME` that requires the customer, their credit terms, and a literal string 'ABC'.

```
EXECUTE PROCEDURE PROCEDURENAME [CUSTOMERMASTER.CUSTOMER], [CUSTOMERMASTER.CREDITTERMS], 'ABC'
```

Understanding the Order of Operations

Ostendo processes all user-defined rules in a specific, fixed sequence. Knowing this order is essential for predicting system behavior and debugging complex scenarios.



1. User Defined Defaults

The system first applies any default values to fields.

2. User Defined Validations

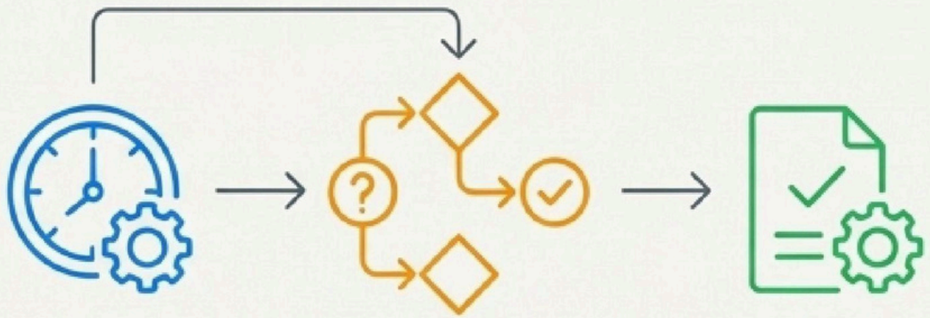
Next, it runs any validation rules to check the integrity of the data.

3. User Defined Actions

Finally, after the data has been defaulted and validated, the system runs your User Defined Actions.

You Have the Blueprint. What Will You Build?

You now have the framework to analyze your business processes and translate them into powerful, automated rules within Ostendo.



WHEN:
Event Occurs

IF:
Condition is Met

THEN:
Action Taken

- What repetitive task could you automate?
- What critical event needs to be logged?
- Where could a timely notification save time or prevent an error?

Start with one small, high-impact rule. The power is in your hands.