

Pre-General Availability: 2024-09-02

# Oracle® Cloud

## Using Robots in Oracle Integration 3



F85348-01  
September 2024

ORACLE®

Oracle Cloud Using Robots in Oracle Integration 3,  
F85348-01

Copyright © 2024, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

# Contents

## Preface

---

|                             |    |
|-----------------------------|----|
| Audience                    | ix |
| Documentation Accessibility | ix |
| Diversity and Inclusion     | ix |
| Related Resources           | ix |
| Conventions                 | x  |

## 1 Start Building Robots Today!

---

## 2 Learn About Robots

---

|   |      |
|---|------|
| Video: Introduction to RPA                                  | 2-1  |
| 5 Reasons to Unify Your Automation                          | 2-1  |
| Create an Automation Strategy                               | 2-3  |
| About Robots  | 2-4  |
| Robots Automate Without APIs                                | 2-4  |
| Differences: Robots and Integrations                        | 2-6  |
| When to Create Robots Versus Integrations                   | 2-6  |
| Available Robot Patterns                                    | 2-8  |
| What Happens When a Robot Runs                              | 2-8  |
| How Robots, Integrations, and Process Applications Interact | 2-9  |
| Building Options for Robots                                 | 2-9  |
| More Robot Concepts   | 2-10 |
| About Projects  | 2-10 |
| About Robot Connections and Robot Connection Types          | 2-10 |
| Predefined Robot Connection Types                           | 2-12 |
| How Robot Connections Work for Multiple Instances           | 2-12 |
| About Environments and Environment Pools                    | 2-13 |
| About the Robot Agent                                       | 2-13 |
| System Requirements   | 2-14 |
| Specifications  | 2-15 |

## 3 Best Practices and Guidelines

---

|  |      |
|--|------|
| Operations Best Practices                        | 3-1  |
| Promote Communication                            | 3-1  |
| Automate Today, Iterate in the Future            | 3-1  |
| Plan Strategically Across Teams                  | 3-1  |
| Understand a Problem Before Solving It           | 3-2  |
| Strive for Continuous Improvement                | 3-2  |
| Environment Best Practices                       | 3-2  |
| Set Up Enough Environments                       | 3-3  |
| Create System Requirements                       | 3-3  |
| Follow All License Agreements                    | 3-3  |
| Replicate the Robot's Environment                | 3-3  |
| Monitor Your Robot Agents                        | 3-3  |
| Maintain Good Performance                        | 3-3  |
| Building and Testing Best Practices              | 3-4  |
| Think Like a Robot, Not a Human                  | 3-4  |
| Define Inputs Using Robot Connections            | 3-4  |
| Future-Proof Your Robots                         | 3-4  |
| Keep the Canvas Tidy                             | 3-4  |
| Create Naming Conventions                        | 3-5  |
| Test in Real-World Conditions                    | 3-5  |
| Use Screenshots to Help with Testing             | 3-5  |
| Processing Bulk Data Best Practices              | 3-5  |
| Questions  | 3-7  |
| Processing Options                               | 3-7  |
| Additional Factors: Number of Robots and Records | 3-8  |
| Sample Calculations                              | 3-9  |
| Simple Scenarios                                 | 3-9  |
| Sequential Processing                            | 3-10 |
| Parallel Processing                              | 3-11 |
| Guidelines for Expressions                       | 3-13 |
| About Expressions                                | 3-13 |
| Mathematical Operators                           | 3-13 |
| Comparison Operators                             | 3-13 |
| Logical Operators                                | 3-14 |
| Inversion Operators                              | 3-14 |
| Functions  | 3-15 |
| Formatting Rules for Expression Syntax           | 3-17 |
| General Formatting Rules                         | 3-17 |
| Formatting Rules for Placeholder Values          | 3-18 |

## 4 Build a Robot

---

|   |      |
|---|------|
| Complete Prerequisites  | 4-1  |
| Review Your Network Configuration                               | 4-2  |
| Create Application Accounts                                     | 4-2  |
| Create a Confidential Application                               | 4-2  |
| Ensure that the Confidential Application is Active              | 4-3  |
| Assign the ServiceDeployer Role to the Confidential Application | 4-4  |
| Meet the Robot Agent's Requirements                             | 4-5  |
| Download the Robot Agent  | 4-5  |
| Update the Robot Agent's Configuration File                     | 4-6  |
| Start the Robot Agent   | 4-8  |
| Start the Robot Agent Automatically                             | 4-9  |
| Install the Recorder  | 4-12 |
| Create a Project  | 4-13 |
| Quick Start for Building Robots                                 | 4-13 |
| Workflow for Planning a Robot                                   | 4-14 |
| Identify the Problem  | 4-14 |
| Understand the Applications                                     | 4-15 |
| Define the Requirements   | 4-15 |
| Workflow for Building a Robot                                   | 4-16 |
| Learn with a Tutorial   | 4-17 |
| 11 Tips for New Robot Builders                                  | 4-17 |
| Tailor Your Building Experience                                 | 4-20 |
| Create Connections to Applications                              | 4-22 |
| Create a Robot Connection Type                                  | 4-22 |
| Create a Robot Connection                                       | 4-23 |
| Create a Robot  | 4-24 |
| Add an Action to a Robot  | 4-26 |
| Add a Checkbox Action   | 4-26 |
| Add a Clear Text Action   | 4-29 |
| Add a Click Element   | 4-30 |
| Add a Click Element Using the Recorder                          | 4-30 |
| Add a Click Element Using the Low-Code Tools                    | 4-33 |
| Add a Close Browser Action                                      | 4-35 |
| Add a Data Stitch Action  | 4-36 |
| Use Case: Assigning a Value                                     | 4-36 |
| Use Case: Appending a Value                                     | 4-36 |
| Add a Data Stitch Action Using the Low-Code Tools               | 4-36 |
| Add a Define Web Table Action                                   | 4-38 |

|   |      |
|---|------|
| Add an Enter Text Action  | 4-41 |
| Add an Enter Text Action Using the Recorder                         | 4-41 |
| Add an Enter Text Action Using the Low-Code Tools                   | 4-43 |
| Add a Get Text Action   | 4-45 |
| Use Cases   | 4-45 |
| Add a Get Text Action Using the Recorder                            | 4-45 |
| Add a Get Text Action Using the Low-Code Tools                      | 4-47 |
| Add a List Action   | 4-49 |
| Use Cases   | 4-49 |
| Add a List Action Using the Recorder                                | 4-49 |
| Add a List Action Using the Low-Code Tools                          | 4-53 |
| Add a Log Action  | 4-57 |
| Add a Login Action  | 4-58 |
| Add an Open Browser Action  | 4-61 |
| Add a Radio Button Action   | 4-63 |
| Add a Screenshot Action   | 4-65 |
| Add a Switch Browser Action   | 4-67 |
| Add a Wait Until Element Is Visible Action                          | 4-69 |
| Add a Wait Until Element Is Visible Action Using the Recorder       | 4-69 |
| Add a Wait Until Element Is Visible Action Using the Low-Code Tools | 4-71 |
| Deeper Dive: Settings for Robot Actions                             | 4-73 |
| Define the Fields of an Action                                      | 4-73 |
| Capture Screenshots in Robots                                       | 4-77 |
| Add Validation to a Robot Action                                    | 4-77 |
| Add Logic to a Robot  | 4-80 |
| Add a Foreach Loop  | 4-80 |
| Add a Switch Condition  | 4-82 |
| Add a Stop  | 4-85 |
| Specify Where a Robot Runs  | 4-86 |
| Understand the Rules for Environments and Environment Pools         | 4-86 |
| Create an Environment Pool  | 4-87 |
| Add Computers to an Environment Pool                                | 4-87 |
| Associate a Robot with an Environment Pool                          | 4-88 |
| Design an Integration That Calls a Robot                            | 4-88 |
| Create and Update a Robot Resource                                  | 4-90 |
| Alternatives to Hard Coding Data                                    | 4-90 |
| Work with Variables   | 4-94 |
| Create a Variable   | 4-94 |
| Update a Variable   | 4-95 |
| Work with a Trigger's Input and Output                              | 4-95 |
| Create a Trigger's Input or Output                                  | 4-95 |
| Update a Trigger's Input or Output                                  | 4-96 |

|  |       |
|--|-------|
| Work with Page States                                  | 4-97  |
| Create a Page State                                    | 4-97  |
| Update a Page State                                    | 4-99  |
| Work with Targets                                      | 4-100 |
| Create a Target  | 4-100 |
| Update a Target  | 4-101 |
| Work with Data Types                                   | 4-101 |
| Create a Data Type                                     | 4-102 |
| Update a Data Type                                     | 4-104 |
| Work with Robot Connections and Robot Connection Types | 4-105 |
| Update a Robot Connection                              | 4-105 |
| Update a Robot Connection Type                         | 4-105 |
| View HTML and XPath                                    | 4-106 |
| View the HTML Code for a Page                          | 4-106 |
| View an Element's XPath                                | 4-106 |
| View All Elements to Target                            | 4-108 |

## 5 Run and Test a Robot

---

|                                  |     |
|----------------------------------|-----|
| Workflow for Testing a Robot     | 5-1 |
| Open a Robot                     | 5-1 |
| Fix a Robot's Errors             | 5-2 |
| Validate a Robot                 | 5-2 |
| Fix an Error in a Robot          | 5-2 |
| Activate a Robot                 | 5-4 |
| Test a Robot on Its Environment  | 5-5 |
| Test a Robot and Its Integration | 5-7 |
| Activate an Integration          | 5-7 |
| Run an Integration and Robot     | 5-7 |

## 6 Troubleshoot

---

|   |     |
|---|-----|
| Troubleshoot the Robot Agent and Environments | 6-1 |
| Cannot Start a Robot Agent                    | 6-1 |
| Troubleshoot the Recorder                     | 6-3 |
| Can't Start the Recorder                      | 6-3 |
| Recorder Suddenly Stops Working               | 6-5 |
| Application Isn't in Browsers List            | 6-6 |
| Canvas Is Read-Only                           | 6-6 |
| Troubleshoot Robots                           | 6-7 |
| Can't Activate a Robot                        | 6-7 |
| Robot Failed                                  | 6-7 |

|  |      |
|--|------|
| Robot Action Takes a Long Time                   | 6-9  |
| Download the Log File for a Robot or Robot Agent | 6-10 |

## A Use Cases

---

|                                     |      |
|-------------------------------------|------|
| Use Case: Update a Set of Invoices  | A-1  |
| Use Case: Save Data After Iterating | A-9  |
| Use Case: Switch Browsers           | A-13 |



# Preface

*Using Robots in Oracle Integration 3* describes how to automate your business processes using robots in Oracle Integration.

## Topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Diversity and Inclusion](#)
- [Related Resources](#)
- [Conventions](#)

## Audience

*Using Robots in Oracle Integration 3* is intended for users who want to create and manage robots in Oracle Integration.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <https://www.oracle.com/corporate/accessibility/>.

### Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <https://support.oracle.com/portal/> or visit [Oracle Accessibility Learning and Support](#) if you are hearing impaired.

## Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

## Related Resources

For more information, see these Oracle resources:

- Oracle Integration documentation on the Oracle Help Center.

## Conventions

The following text conventions are used in this document.

| Convention      | Meaning  |
|-----------------|--|
| <b>boldface</b> | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.         |
| <i>italic</i>   | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.                          |
| monospace       | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

# 1

## Start Building Robots Today!

Robotic process automation (RPA) is available directly within in Oracle Integration. Adopters of this exciting new technology can start building robots today.

- If your organization already uses Oracle Integration, enter a service request (SR) to start using robotic process automation.
- If your organization doesn't yet use Oracle Integration, [contact Oracle to set up a demo](#). You can also [get a free trial of Oracle Integration](#).

Before you start building robots, review your use cases and requirements with your contact at Oracle to ensure their feasibility.

# 2

## Learn About Robots

Robots and integrations help you achieve the same business goals using different technologies. A robot performs UI-based automation, whereas an integration performs API-based automation. You can build both robots and integrations using Oracle Integration.

### What Do You Want to Do?

| Goal  | Links   |
|---|---|
| Watch a quick introduction to robots                    | <a href="#">Video: Introduction to RPA</a>  |
| Plan how to fit robots into your business               | <a href="#">5 Reasons to Unify Your Automation</a><br><a href="#">Create an Automation Strategy</a> |
| Learn about robots and robotic process automation (RPA) | <a href="#">About Robots</a><br><a href="#">More Robot Concepts</a>                                 |

## Video: Introduction to RPA

Watch a video that introduces the robotic process automation (RPA) capabilities in Oracle Integration.



## 5 Reasons to Unify Your Automation

With Oracle Integration, your organization can automate efficiently and effectively by producing UI-based automation and API-based automation in one complete automation platform.

If your organization already has a team or department that delivers robotic process automation (RPA), that's great! The RPA capabilities in Oracle Integration can still help your business. Keep reading to learn more.

### 1. Bridge the Gaps Between Your Automation Strategies

Many organizations divide their automation work and strategies. For example:

- IT teams perform API-based automation by designing integrations.  
They can't automate legacy applications that don't have APIs because their tools don't allow them to develop UI-based automation.
- Business teams perform UI-based automation by building robots.  
They can't develop more complex automation because their tools don't allow them to develop API-based automation.

Both teams have the same goals: automating their business processes. However, their separate tools lead to automation gaps, including imperfect and abandoned automation efforts.

With Oracle Integration, organizations can unify their automation work and their automation strategies. IT and business teams can collaborate on this automation work. Individuals with

more technical skill sets can work with APIs and more complex integrations, and people with strong business acumen can work with robots.

When you can perform UI-based automation and API-based automation in one place, you can find the right solution for the problem that you're facing and eliminate your automation silos.

See [Create an Automation Strategy](#).

## 2. Ensure Good Corporate Governance

Teams typically delight in the opportunity to streamline tedious and time-consuming work through automation. However, unfettered access to automation tools can lead to challenges that impact your key business systems. For example, a large number of people working in an application at the same time can impact its performance, and the same is true for robots. Similarly, a large number of robots could infringe upon license agreements. Good governance is critical to such situations.

With Oracle Integration, you gain the efficiencies of automation along with the controls to guarantee good corporate governance. A central team can vet all new robots to ensure that they have a positive impact on your business operations. They can also design the integrations that call the robots and deploy the robots and integrations to production.

With this controlled process, everyone wins. Business teams can automate their business processes with only minimal involvement from IT, while a designated team maintains operational excellence by providing oversight.

## 3. Manage the Lifecycle of Assets in One Place

Oracle Integration offers a central place manage the lifecycle of your assets from development to deployment.

Whether you're developing a new robot or integration or updating an existing robot or integration, you work in one place. And when you're ready to promote your work from development to production, you don't need to go to multiple applications and find all the pieces and parts that you need and move them over individually. Instead, deploy your related robots and integrations together as a single automation solution.

Gain efficiency and reduce risk by managing the entire lifecycle of your automation assets in one place.

See [Get Started with Projects in \*Using Integrations in Oracle Integration 3\*](#).

## 4. Unify Observability

When your organization works in several automation tools, your monitoring work can be inefficient and inconsistent. For example:

- There is no central place to monitor all automation.
- Teams must access separate dashboards and monitoring tools.
- Teams must create and follow different procedures for monitoring and responding to issues.

Oracle Integration addresses these issues by offering unified observability of all automation assets. Gain efficiencies by monitoring everything in one place with robust observability tools, including dashboards, proactive email notifications, and detailed troubleshooting pages.

See [Workflow for Monitoring Integrations and Get Started with Observability in \*Using Integrations in Oracle Integration 3\*](#).

## 5. Build Flexible and Future-Proof Solutions

No matter how you're automating your business, you always start with an integration. For example, even when you need to do UI-based automation by building a robot, you must also design an integration that calls the robot.

This construct helps future proof your work. For example, consider a scenario in which your business requirements and automation strategy require a robot. You create an integration that calls the robot and then move on to your next automation work. Then, if your requirements change in the future and you need to replace the UI-based automation with API-based automation, you can make the update without impacting your business process. Simply version the integration and replace the robot with an API-based integration.

### Learn More

To understand the value of robots, see [About Robots](#).

# Create an Automation Strategy

With Oracle Integration, you can automate any application, whether it has APIs or not. Combine your new enterprise-level perspective on automation with an automation strategy to ensure that your automation decisions are future proof.

## What is an Automation Strategy?

An automation strategy specifies your overall plan for automating your organization's processes. For instance, an automation strategy might answer the following questions:

- How do you determine the business processes to automate?
- What analysis do you complete before and after automating a process?
- How do you improve a business process after automating it?
- How do you manage the life cycle of your automation?
- How do you manage the environments where robots run?
- When do you use API-based automation, and when do you use UI-based automation?

### Note:

This guide refers to the process of creating robots and integrations as **automating**, whether the robots and integrations automate or digitize work. For example, an integration might send a notification email for an approval, and a person must still approve the request. Such an integration digitizes some of this work.

## Sample Automation Strategy

Your automation strategy can be high level, detailed, or anywhere in between. Consider the following high-level strategy as a starting place.

- Our organization values rapidly digitizing and automating our current business processes over replanning the processes.
- We believe that our automation journey begins, rather than ends, when we automate and digitize our business processes.

- We develop API-based automation when feasible and UI-based automation when necessary.
- We strive to continuously improve the efficiency and effectiveness of our business processes by analyzing the insights that we gain from automating and digitizing them.

### Expect Your Automation Strategy to Evolve

Consider a situation in which a business team has historically developed UI-based automation, while an IT team has developed API-based automation.

With Oracle Integration, when the IT team's automation requires UI-based automation, they now can build a robot. Similarly, when a business team's automation benefits from API-based automation, IT teams now can contribute to the solution. Teams can build confidently and work together in Oracle Integration, knowing that they're delivering a fast, flexible, and future-proof solution.

Whether you manage this automation work centrally within your organization or locally within teams is up to you. As you automate more and more solutions using Oracle Integration, you might find that your automation strategies and the way that you manage your automation solutions also continue to evolve.

## About Robots

A robot is a software script. A robot specifies the steps to follow for accomplishing a task using an application's user interface. A robot performs work just like a human or an integration, and it helps your business become more autonomous.

### Topics:

- [Robots Automate Without APIs](#)
- [Differences: Robots and Integrations](#)
- [When to Create Robots Versus Integrations](#)
- [Available Robot Patterns](#)
- [What Happens When a Robot Runs](#)
- [How Robots, Integrations, and Process Applications Interact](#)
- [Building Options for Robots](#)

## Robots Automate Without APIs

Need to automate a business process, but APIs don't expose the fields you need to update? A robot can help!

### How Robots Fit into a Broader Automation Strategy

A robot is another tool in your toolbox for automating your business processes.

With robots and integrations, you can automate *everything*, from robust enterprise software with extensive APIs to legacy systems without any APIs.

Your automation work fits into one of the following categories:

- **UI-based automation**

When you're automating an application that doesn't have APIs for your specific business process, build a **robot**, and specify the workflow from within the application user interface.

- **API-based automation**

When you're automating an application that has APIs for your specific business process, design an **integration**, and call the APIs from the integration.

If you have business processes that you've been unable to automate because the applications don't have APIs, robots are a game changer.

### **Already Familiar with Integrations?**

If you've built integrations before, you're already an expert in robots. Robots and integrations provide two ways to automate your business and have many commonalities. For example:

- You build integrations and robots in projects.
- The canvases where you build integrations and robots have the same look and feel.
- Integrations and robots both require connections to connect to applications.
- You deploy integrations and robots the same way.
- The observability pages are the same for integrations and robots.

Additionally, robots and integrations complete the same type of work, including the following:

- Automating your business processes by connecting to applications, getting information from the applications, and passing the information back to Oracle Integration.
- Improving the efficiency and effectiveness of your business and offering valuable insight into your business processes.

### **When to Build a Robot**

When creating a robot, you specify the steps that a human takes when interacting with a user interface. For example, open a browser, enter data into a field, and click a button.

Build a robot when an application doesn't have APIs, when the APIs are inaccessible, or when integration development resources aren't available.

See [When to Create Robots Versus Integrations](#).

### **Robots Require Integrations**

In Oracle Integration, automation begins with an integration. Even if you're automating a business process with a single robot, the only way to run the robot is to call it from an integration.

While this approach might seem like extra work, the work is an investment in the future of your business processes. For example:

- Integrations provide robust observability capabilities to monitor your operations
- Consider a scenario in which a robot creates an order and a process application calls the robot.

In this scenario, if the robot no longer meets your organization's needs and needs to be reconfigured as an integration, you also must update the process application to call the integration instead. Updating your process application when your business needs change can lead to interruptions in your business processes.

Instead, with Oracle Integration, the process application calls an integration, and the integration calls a robot. If you need to switch from UI-based automation to API-based automation, all you need to do is modify the integration so that it calls an integration instead of a robot.



## Enjoy Simplicity While Gaining Control

Most robots in the automation industry record one path through a user interface. If you want the robot to take a different path, you must build another robot. Before you know it, you've built an army of robots that can complete only very specific tasks and can't do anything when the application that they automate goes down.

Robots in Oracle Integration are different. They combine the simplicity of a recording with the control of a low-code environment. To create a recording, simply work in an application as users normally do. Oracle Integration generates the code for this automation as you work, providing you with valuable knowledge of how the automation code works. As you work, tweak the automation and define parameters for the automation using an intuitive low-code editor.

When you finish creating the recording, you've also finished your automation work. Instead of creating an army of robots, you've created one robot that can handle numerous use cases.

Because you call every robot from an integration, you can incorporate fault handlers and error management to gracefully handle all the situations that the robot might encounter. And, thanks to consistent user interfaces and terminology between integrations and robots, integration developers typically can learn how to build robots very quickly.

Even better? You build robots from within Oracle Integration, without needing to install a special additional application.

## Differences: Robots and Integrations

Robots and integrations provide similar capabilities, including automating your business by connecting to applications and sending data back to Oracle Integration. However, robots and integrations get information from applications in different ways.

| Area   | Integrations   | Robots  |
|--|--|---|
| <b>How it gets information from an application</b> | An integration gets information from an application by calling its APIs.   | A robot gets information from an application by opening the application and completing a task in the user interface as a person does.                                     |
| <b>Key requirements</b>                            | An integration developer with a broad and deep technical skill set designs an integration.<br>The integration needs access to an application's APIs.   | A robot developer who is comfortable with technology builds a robot.<br>The robot needs access to an application's user interface.  |
| <b>Sample usage for an ERP system</b>              | When you want to automate the creation of orders and APIs are available for the work, you can design an integration that connects to the ERP application's APIs. You connect to the APIs using an adapter in Oracle Integration. | When you want to automate the creation of orders and an API isn't available for the work, you can build a robot that interacts with the ERP application's user interface. |

## When to Create Robots Versus Integrations

You have several options for automating your business, including integrations and robots. Integrations use API-based automation, and robots use UI-based automation. Learn when to

use each type of automation so that your organization develops the right automation for each use case.

### When to Create an Integration

When automating a business process, if an application supports an integration, you should design an integration. Integrations offer the most scalable and robust automation solutions.

When all of the following statements are true, design an integration:

- The applications that you're automating have APIs.
- The APIs can access and update the fields that the business process uses.
- You have access to all of the APIs.
- Your organization has the staffing resources to design an integration.

### When to Create a Robot

You typically build a robot when one or more blockers prevent you from designing an integration. For example, you build a robot under the following circumstances:

- The applications that you're automating have no APIs; or their APIs can't access and update the fields that the business process uses.
- The applications that you need to automate have APIs, but operational or logistical challenges prevent you from accessing them. For example:

- The APIs are currently inaccessible.

For example, consider an application for which the APIs must be enabled. The team that manages the application is willing to enable the APIs, but only when they perform routine maintenance on the application. The next maintenance period is nine months away.

- Integration developers are currently unavailable.

Developing integrations and robots is intuitive and straightforward, and often times, integration developers and robot developers can do the same work. However, sometimes no one is available for the integration work.

In both cases, you could postpone the automation work until your organization can support the design of an integration. Or, you can eliminate these bottlenecks and automate your business now by rapidly prototyping a robot.

In the future, when the APIs are accessible and the integration developers have availability, you have the option of replacing the robot with an integration and incorporating more robustness and scalability into the automation. Or, you might find the robot suits your business needs.

**Reminder:** Even when you build a robot, you still need to design an integration that calls the robot.

#### Tip:

Your decision about creating an integration or a robot is important and worthy of careful consideration, but you can always change your mind. With Oracle Integration, you can easily switch a robot with an integration, or an integration with a robot, without impacting your business process in any way.

## Available Robot Patterns

Robots interact with user interfaces just as humans do. Robots can complete the manual tasks that humans have historically had to do.

| Robot pattern          | Description  | Sample use cases   |
|------------------------|--|--|
| Web-based applications | <p>You can build a robot that interacts with any web-based application, such as a website or an application that you access from an internet browser.</p> <p><b>Note:</b> You currently cannot run a robot on a page that contains an inline frame (iframe). If you run the recorder on a page with an iframe, the recorder informs you that iframes aren't supported.</p> | <p><b>Balancing invoices</b></p> <p>An integration identifies the invoices that are out of balance, but APIs are not available to balance the invoices. You can build a robot to open each out-of-balance invoice, compare specific values, and update the invoice as needed so that it's in balance. The finance team that has historically needed to complete these tasks is therefore relieved of this manual and tedious work.</p> <p><b>Getting the supplier name from an invoice</b></p> <p>A business application requires information from an invoice. An integration can collect most information, but APIs aren't available to obtain the supplier name. You can build a robot to get the supplier name, and an integration can pass this information to the business application. The finance team no longer needs to complete this manual work.</p> <p><b>Updating emergency contact information</b></p> <p>An employee provides various information as part of their onboarding. APIs can send most of the data to your HCM software, but APIs aren't available for sending the emergency contact information. You can build a robot to update the emergency contact so that your human resources team doesn't need to complete this manual task.</p> |

For more information about use cases, see [Use Cases](#).

## What Happens When a Robot Runs

Understanding the events that occur when a robot runs can provide helpful context for your setup work.










You might spend a minute or two reading this information, but keep in mind that most of these tasks typically finish in seconds or less.

| Order | Task  | Details   |
|-------|---|---|
| 1     | An integration runs and kicks off the robot       | An integration runs according to its logic. When the integration reaches the point in its flow where it invokes the robot, the integration notifies Oracle Integration that the robot needs to run.                 |
| 2     | Oracle Integration chooses where to run the robot | Oracle Integration finds the environment pool that you configured the robot to run on.<br>Next, Oracle Integration looks within the environment pool and selects an environment that is available to run the robot. |

| Order | Task   | Details  |
|-------|--|--|
| 3     | The environment collects the latest version of the robot | The robot agent that is installed on the selected environment detects that a robot instance needs to run. The robot agent gets the latest version of the robot and all connection details from Oracle Integration and runs an instance of the robot. |
| 4     | A robot instance runs on the environment                 | The robot agent runs the robot instance on the environment. The robot agent sends regular status updates back to Oracle Integration.   |
| 5     | The robot instance finishes running                      | The robot instance finishes running and reports its status to Oracle Integration. Oracle Integration shares this status information with the integration that invoked the robot and allows the integration to continue to its next step.             |

## How Robots, Integrations, and Process Applications Interact

Robots, integrations, and process applications interact in different ways. For instance, an integration can call a robot, but a robot cannot call an integration or another robot.

| Type of automation  | Can call an integration?  | Can call a process application?   | Can call a robot?   |
|---------------------|---|---|---|
| Integration         |    |    |    |
| Process application |   |   |   |
| Robot               |  |  |  |

## Building Options for Robots

To build a robot, record the actions that a robot completes, similar to a screen recording. At any time, you can pause the recording and harness the control and power of the low-code capabilities, which let you drag actions to the canvas in Oracle Integration.

### Quick Introduction

The recorder is the building tool of choice for most robot developers because it's fast and easy, and all the control of the low-code capabilities remains at your fingertips.

### Start and Stop Anywhere, and Update as You Build

You can start and stop the recorder and the low-code capabilities from anywhere in your robot, so you can always use your tool of choice to build.

Similarly, both the recorder and the low-code capabilities generate code as you work, and you can update the code as needed right away. Updating as you build offers several benefits. You're engaged in the code generation from the beginning, and you can swap out hard-coded values for parameters as you work.

### Learn More

You don't need extensive programming knowledge to use either building tool. Give them both a try! See [Quick Start for Building Robots](#).

## More Robot Concepts

Familiarize yourself with the components that are related to robots.

### What Do You Want to Learn About?

- [About Projects](#)
- [About Robot Connections and Robot Connection Types](#)
- [About Environments and Environment Pools](#)
- [About the Robot Agent](#)

## About Projects

In a project, you build your automation solutions, manage their deployment, and observe them.

### New to Projects?

Projects are the hubs of all your automation work and the place to go when you want to design automation solutions, including integrations and robots. Each project can focus on a specific business objective. People who are on different teams and who have different skill sets can collaborate in the project to meet the objective.

Projects provide convenient deployment and unified observability, allowing everyone to work together to build, deploy, and monitor integrations and robots.

To learn more about projects in general, see [Get Started with Projects in \*Using Integrations in Oracle Integration 3\*](#).

### Familiar with Projects?

If you're already familiar with projects, here are some key points to know about how UI-based automation and robots work in projects:

1. You create and manage the following components in a project:
  - Robots
  - Robot connections and robot connection types
  - Environment pools
  - Integrations that call robots
2. When testing, you can run robots and the integrations that call robots from within a project.

## About Robot Connections and Robot Connection Types

A robot connects to an application using a robot connection. You specify the parameters that you define in a robot connection using a robot connection type.

### Familiar with Connections for Integrations?

Are you already familiar with connections for integrations? Connections for robots are similar. An integration connection and a robot connection both connect to an application. However, they have some key differences, which the following table explains.

| Area   | Integration connections  | Robot connections   |
|--|--|---|
| <b>What the connection is based on</b>               | An integration connection is based on an adapter.<br>An adapter is specific to an application or technology and includes information about the connection type and security protocols. | A robot connection is based on a robot connection type. You can use the predefined robot connection types, or create your own.<br>A robot connection type defines how to connect to an application, such as by specifying a URL and user credentials. |
| <b>How the connection connects to an application</b> | An integration connection uses Java code to call an API.<br>For example, the integration connection might perform an invoke activity for the application that it's connecting to.      | A robot connection contains the information that a robot needs to open an application.  |
| <b>How to test a connection</b>                      | After creating a connection, you can test it to ensure that you configured it correctly.   | You test a robot connection as part of testing a robot.   |

Keep reading to learn more.

### Robot Connection Types Are Similar to Templates

Unlike an integration, a robot doesn't need information about an application's security protocols or its APIs. A robot typically needs only a little information about the application or web page that it's connecting to, such as a URL and credentials. You list the fields that a robot needs to connect to an application in a robot connection type.

A robot connection type doesn't list the values of the fields, and it's not application specific. Therefore, you can base robot connections to different applications on the same robot connection type. For example, any robot connection that requires only a user name, password, and URL to access an application can use one of the predefined robot connection types.

To save you some time, Oracle provides the most commonly used robot connection types. See [Predefined Robot Connection Types](#). If an application requires parameters that aren't included in the predefined robot connection types, you can create a new robot connection type.

### Robot Connections Are Based on Robot Connection Types

You specify the parameters that a robot needs in a robot connection type, and you specify values for the parameters in a robot connection. For example, a user name is a parameter; jane.doe@example.com is an example of a value of this parameter.

You base every robot connection on a robot connection type.

### Separation Provides Benefits

A robot connection is separate from a robot, and this separation offers several benefits. For example:

- Robot connections protect secret information.  
You store secret information, such as credentials, in a robot connection. Robot connections are stored with an extra measure of security, and their information doesn't appear in the activity stream.
- Reusing robot connections makes updates faster.

When your organization rotates passwords, you can update a password in one place, and all robots that use the connection start using the new password. Reusing connections makes for faster, easier, and less risky updates.

- Multiple robots can use the same robot connection, so you can build robots more quickly.
- You can update a robot connection of an active robot, without having to deactivate it first.

Therefore, you can quickly update a robot after promoting it to a different environment.

## Predefined Robot Connection Types

When you create a robot connection, you can base it on a predefined robot connection type that specifies commonly required connection parameters. Or, if your application requires different parameters, you can base a robot connection on a robot connection type that you create.

The following predefined robot connection type is available. If you need to create your own robot connection type, see [Create a Robot Connection Type](#).

| Name                 | Parameters   | Usage  |
|----------------------|--|--|
| Oracle RPA web login | <ul style="list-style-type: none"> <li>• User name</li> <li>• Password</li> <li>• URL</li> </ul> | Signing in to a web application that requires only a user name and password. |

## How Robot Connections Work for Multiple Instances

You might have multiple instances of Oracle Integration and of the applications that your robots connect to. Learn how to optimize your workflow for each scenario.

### Multiple Oracle Integration Instances

| Factor to consider  | More information  |
|---|---|
| Building an automation solution in a development environment      | You can build and test your robot in one or more lower Oracle Integration environments without impacting your production environment.   |
| Deploying an automation solution to a higher environment          | Use the built-in capabilities in a project to seamlessly deploy an automation solution, including robots and integrations, to a higher environment, such as a production environment.<br><br>When you promote a solution to a higher environment, the values in the integration and robot connections are removed. For security reasons, Oracle Integration doesn't include sensitive data, such as passwords, in deployment packages. You likely need to provide different values for a higher environment, anyway. For example, the robot must work in a production instance, which might have a different URL and different credentials. |
| Updating an automation solution when a robot's credentials change | The password that a robot uses is sensitive information. Often, one person builds a robot and another person enters these credentials.<br><br>When a robot's password changes, only the robot connection needs to be updated. You don't need to open or update the robot to make this change.<br><br>If a robot has already been deployed to a production environment, you can update the password directly in the production environment.  |

## Multiple Instances of the Application That a Robot Automates

| Factor to consider   | More information   |
|--|--|
| Building and testing an automation solution against a development instance | While you're building and testing a robot and its integration, point to the development instance of the application.<br>Identify any differences between the development and production instances of the application. These differences could impact the robot's behavior in production. For instance, sometimes new features are deployed to a development environment first. |
| Pointing to a higher environment after deployment to production            | After you deploy an automation solution to a production environment of Oracle Integration, update the connections so that they point to the production instance of the application.  |

## About Environments and Environment Pools

An environment is the computer or virtual machine where instances of your robots run. An environment pool is similar to a cluster and is a collection of computers or virtual machines that can run specific robots. You must associate each environment with exactly one environment pool.

### Your Requirements Determine the Environments You Need

Knowing the time a robot needs to run is the key to setting up environments effectively and efficiently.

For instance, if a robot takes one minute or less to run, and you need to run one transaction per minute, you typically need only one environment.

However, if you need to run two transactions per minute, you need two environments, or you'll end up with a bottleneck situation and delayed transactions.

### Your Responsibilities

Your organization is responsible for the following tasks:

- Setting up and managing the environments that you need to run your robots.  
An environment is typically a Windows virtual machine (VM). You can also use a Mac or Linux machine.  
Depending on your automation requirements, you might build several VMs or hundreds of VMs for robots to run on.
- Associating each environment with exactly one environment pool.  
For example, you might have an environment pool for human resources robots, another for finance robots, and so on.
- Installing and managing the robot agent on each environment.  
See [About the Robot Agent](#).

## About the Robot Agent

You install a robot agent on every environment, which is the computer or virtual machine where a robot runs. The robot agent runs the instances of the robots on these environments.



To learn more about environments, see [About Environments and Environment Pools](#).

### Familiar with the Connectivity Agent?

If you're familiar with the connectivity agent in Oracle Integration, the robot agent offers similar capabilities with some differences.

| Area                           | Connectivity agent  | Robot agent   |
|--------------------------------|---|---|
| <b>Usage</b>                   | You use the connectivity agent when an integration must connect to a private or on-premises network.  | You use the robot agent to run a robot on an environment.                           |
| <b>Installation and number</b> | You install the connectivity agent on an on-premises machine, typically a virtual machine.<br><br>Your organization can install one or many connectivity agents.  | You install the robot agent on every environment that will run a robot.             |
| <b>Components</b>              | The connectivity agent actually includes two agents: <ul style="list-style-type: none"><li>• SaaS agent, which runs in Oracle Integration. Oracle manages this agent.</li><li>• On-premises agent, which you must install and manage.</li></ul> | The robot agent includes only the robot agent that you install on each environment. |


### Your Responsibilities

Your organization is responsible for the following tasks:

- Ensuring that all environments meet the system requirements for the robot agent.  
See [System Requirements](#).
- Installing the robot agent on every environment and, if required, configuring the robot agent as a Windows service.  
See [Complete Prerequisites](#).
- Keeping the robot agent up to date on every environment.

## System Requirements

You must install the robot agent on every environment, so you must ensure that every environment meets the system requirements for the robot agent.

| Area     | Requirements   |
|----------|--|
| Software | <ul style="list-style-type: none"> <li>• Java Development Kit (JDK) 17</li> <li>• One of the following browsers for the environment where the robot agent runs: <ul style="list-style-type: none"> <li>– Apple Safari</li> <li>– Google Chrome (recommended) or Headless Chrome</li> <li>– Microsoft Internet Explorer</li> <li>– Microsoft Edge</li> <li>– Mozilla Firefox or Headless Firefox</li> </ul> </li> </ul>   |
|          | <div style="border: 1px solid #0070C0; padding: 10px; background-color: #E6F2FF;"> <p> <b>Note:</b></p> <p>Oracle Integration has different browser requirements. See Supported Browsers in <i>Getting Started with Oracle Integration 3</i>.</p> </div>  |
|          | <ul style="list-style-type: none"> <li>• One of the following options for storing secret information in the computer's local keystore: <ul style="list-style-type: none"> <li>– Linux and Unix computers: Keyutils<br/>You can install Keyutils by installing Homebrew, Yum, or AptGet.</li> <li>– Windows computers: PowerShell<br/>This solution is usually included in the operating system of Windows computers.</li> <li>– Mac computers: Security Utility<br/>This software is usually included in the operating system of Mac computers.</li> </ul> </li> </ul> |

## Specifications

Get an in-depth understanding of the robot agent by reviewing its communication method, security protocols, and more.

| Area                    | More information  |
|-------------------------|---|
| Communication and ports | <ul style="list-style-type: none"> <li>• The robot agent initiates all communication and must be able to contact Oracle Integration so that it can poll for work.<br/>See <a href="#">Review Your Network Configuration</a>.</li> <li>• No ports are opened on the on-premises system for communication.</li> </ul>   |
| Security protocol       | <ul style="list-style-type: none"> <li>• The robot agent registers with Oracle Integration over SSL using the provided Oracle Integration credentials.<br/>See <a href="#">Complete Prerequisites</a>.</li> <li>• All communication is secured using SSL.</li> <li>• The robot agent processes requests by pulling messages from Oracle Integration across SSL.</li> <li>• The robot agent posts responses by pushing messages to Oracle Integration across SSL.</li> </ul> |
| Data persistence        | No data is persisted in the robot agent.  |

# 3

## Best Practices and Guidelines

Spend some time reviewing the best practices, workflows, and guidelines so that you can build like an expert from the beginning.

### Where Do You Want to Start?

- [Operations Best Practices](#)
- [Environment Best Practices](#)
- [Building and Testing Best Practices](#)
- [Processing Bulk Data Best Practices](#)
- [Guidelines for Expressions](#)

## Operations Best Practices

### Promote Communication

At many organizations, one person or team builds a robot, and another person or team designs the integration that calls the robot. Sometimes a third person or team is responsible for monitoring the automation after it goes live.

In such cases, communicate your goals and timelines early and often. Identify people's availability, create a schedule for the work, and schedule periodic check-ins to ensure that everyone is working toward the same deadlines.

### Automate Today, Iterate in the Future

If today's business process is working for your organization, you don't need to redesign the process before you can automate it. Automate it today, and use the insights that you gain to find opportunities for efficiency and effectiveness.

For more details, see [Identify the Problem](#).

### Plan Strategically Across Teams

At many organizations, different people and sometimes different teams are responsible for building robots and integrations. Oracle Integration supports this collaborative work by allowing integration and robot developers to complete their work in parallel or sequentially, all while collaborating within a project.

While planning and working, follow the best practices for the development of component-based software. Work intentionally by defining the contracts up front, and make sure everyone agrees to them. For example, a contract might state that a robot expects to get a purchase order and deliver a supplier name.

After everyone has agreed to the contracts, work can begin. A robot developer can build and test a robot, even if an integration developer isn't available to start work on the integration yet. Similarly, an integration developer can start working after a robot developer creates only the shell of a robot, a task that takes just a couple minutes. See [Create a Robot](#).

## Understand a Problem Before Solving It

It's important to fully understand the problem you're trying to solve before you start trying to plan a solution.

All too often, software has limitations and restrictions. Because you know your applications so well, you might focus on what your software tools let you do rather than on your business goals. As a result, you could start making compromises from the beginning. Your solution becomes what your software can support, not what your business requires. The end result is often a solution that solves only part of the problem.

With Oracle Integration, you don't need to make compromises on automation because you can design API-based automation and UI-based automation in the same place.

Oracle meets you where you are and helps make your applications more intelligent. With Oracle Integration, you can create a complete, simple, observable, and intelligent automation strategy.

## Strive for Continuous Improvement

Deploying your automation solution to production is, in many ways, the beginning of your automation journey. After deploying a solution, you can analyze and optimize it.

After you deploy an automation solution, analyze your solution, and look for opportunities for tactical improvements that move your organization toward a more perfect business process.

Remember that automation offers three key benefits:

- **Efficiency:** For example, fulfilling the same number of orders using fewer resources.
- **Effectiveness:** For example, fulfilling orders with fewer errors and in less time.
- **Insight:** For example, identifying the issues that require more resources to fulfill orders, the issues that lead to errors, and the issues that lead to order delays.

The insights that you gain from your automation are the keys to optimizing the automation. These insights help you identify the changes that you need to make.

For example, after you automate an approval process, you might see that a team requires 6.5 days on average to offer their approval. This delay leads to inefficiencies, downstream delays, and stress. The insight and an understanding of its business impact help you determine next steps, such as:

- Expanding the pool of approvers
- Removing the approval requirement
- Identifying a daily designated approver who must approve all open requests by the end of each day

## Environment Best Practices

## Set Up Enough Environments

Make sure that you've created enough environments to handle the workload of your robot.

For example, consider a robot that runs in two minutes and has one environment. If you call the robot every three minutes, the robot typically runs without issues, even with occasional performance issues. However, if you call the robot every minute, you create a backlog.

How do you know how many environments to run? Consider the following factors:

- Amount of time a robot instance typically requires to run
- Frequency with which the robot is called
- Likelihood of performance issues
- Impact to the business if a backlog occurs

## Create System Requirements

Each robot has different requirements. Document the requirements and use them when setting up and testing each environment. Make sure to include all software requirements, configuration requirements, and network connectivity requirements.

## Follow All License Agreements

Verify that your applications' license agreements allow for automation.

## Replicate the Robot's Environment

Your work computer likely has numerous time-saving features, including applications that remember your credentials and browsers that remember your preferences. However, a robot's environment is likely to be a clean slate.

Additionally, the account that you use to record a robot could inadvertently have more or fewer permissions than the robot will have in production.

To ensure that your robot can run as expected in a production environment, make sure that the experience you have while building a robot accurately replicates the experience that the robot will have while running on its environment.

## Monitor Your Robot Agents

Periodically throughout the day, monitor your robot agents. For example, ensure that all expected agents are up and running.

For more guidance on monitoring, see *Workflow for Monitoring Integrations* in *Using Integrations in Oracle Integration 3*

## Maintain Good Performance

A robot works in an application like a person does. Before creating your robots, understand the performance implications.

For example, consider a scenario in which ten robots are active in an application at a time. Determine whether the robots could impact the performance of the application, and take the

appropriate action. For instance, you could reduce the number of robots, run the robots overnight, or increase the capacity of the machine that hosts the application.

## Building and Testing Best Practices

### Think Like a Robot, Not a Human

In an ideal world, a robot completes the same tasks every day without interruption or fail. However, in the real world, robots sometimes get stuck.

Robots typically get stuck due to the following reasons:

- A robot doesn't understand an element in a user interface (UI).  
How to plan for this scenario: Remember that a robot interacts with an application differently than a human does, and build your robot intentionally for the robot's requirements. For instance, humans typically interact with software using their vision. When people encounter issues, they often click around and troubleshoot to find a way forward. Robots don't have these abilities. A robot can do only what it's programmed to do, nothing more or less. While building a robot, thinking about its actions programmatically, rather than visually, helps you build more effectively.
- The UI or HTML for a page changes.  
How to plan for this scenario: The robots in Oracle Integration are continually evolving to better handle these scenarios. Additionally, you can use global targeting in your actions so that you can more quickly update fields when needed.

### Define Inputs Using Robot Connections

When defining an action's input, such as the URL for a browser, use the values that you defined in a robot connection whenever possible.

Specifying a value one time in the robot connection and pointing to that variable lets you isolate this information. When you need to update the value in the future, you need to update only the robot connection, rather than every robot that uses the connection.

### Future-Proof Your Robots

Don't hard-code data when you don't need to. Instead, future-proof a robot by using placeholder values.

See [Alternatives to Hard Coding Data](#).

### Keep the Canvas Tidy

You build your robot in the canvas. Keep the canvas organized so that you can find information quickly.

For example:

- Use clear names for actions.
- Group related actions into a step group.
- Use clear names for step groups.

For more guidance, see [Quick Start for Building Robots](#).

## Create Naming Conventions

When performing UI-based automation, you must provide names for a variety of components, including robots, the environments that the robots run on, and the actions that the robots perform.

Some objects, such as robots, connections, and environment pools, can include spaces. Other more programmatic objects, such as variables and input parameters, cannot include spaces.

If your organization doesn't have them yet, create naming conventions. Then, follow the conventions so that everyone understand the components at a glance and update a robot quickly and confidently.

## Test in Real-World Conditions

Test a robot early and often using real-world conditions.

See [Run and Test a Robot](#).

## Use Screenshots to Help with Testing

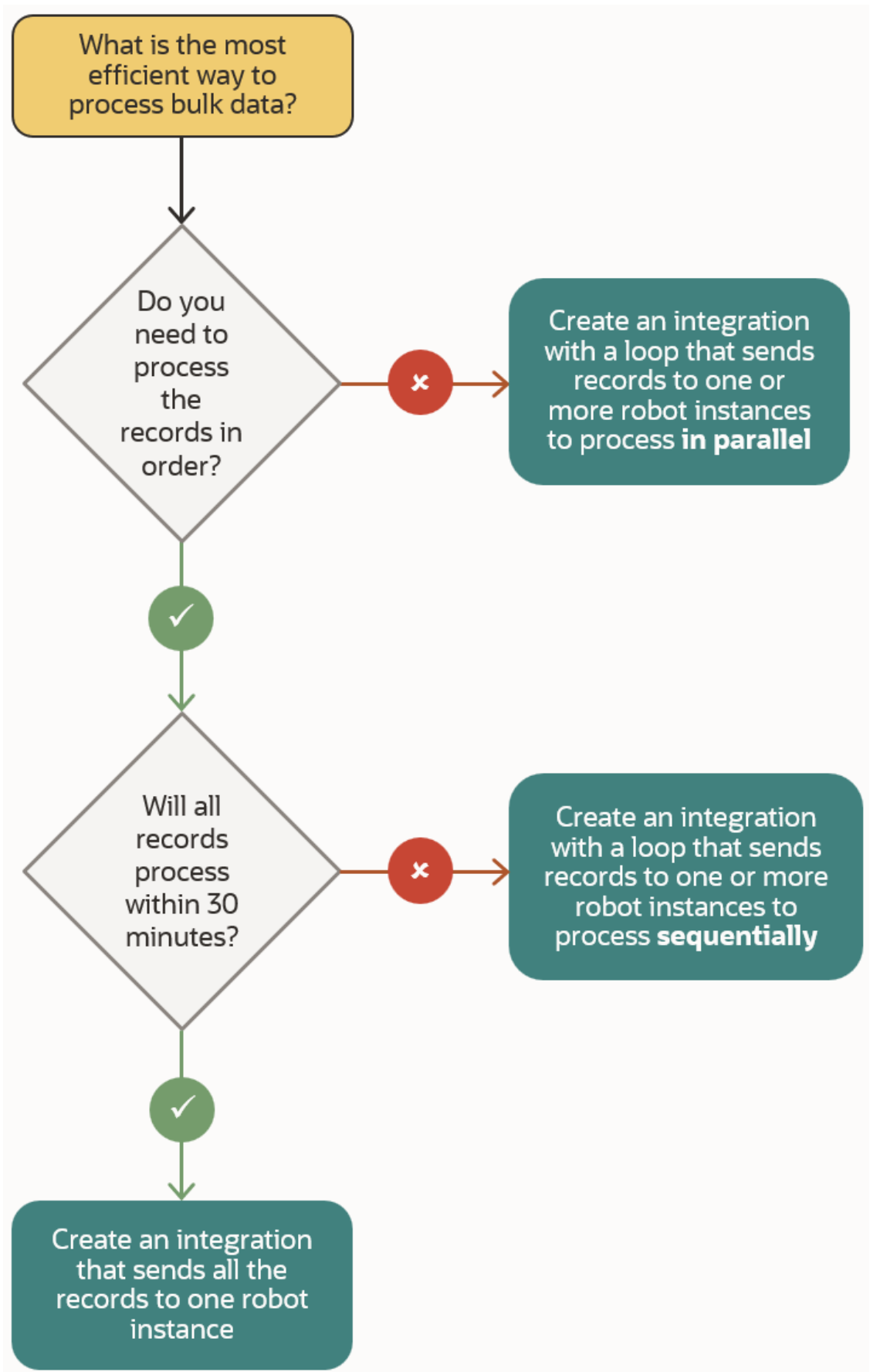
When you add an action to a robot, you have several options for capturing screenshots, including capturing a robot's view before and after completing an action. The screenshots are helpful when troubleshooting a failed robot instance.

For example, if a robot's credentials don't have the appropriate access, a robot might not be able to see the field it needs to update. A screenshot of the missing field can help you quickly troubleshoot any issues that the robot encounters.

See [Capture Screenshots in Robots](#).

## Processing Bulk Data Best Practices



Your automation work might involve processing bulk data, such as a file of input data or a JSON object containing a number of items. You have several options for processing bulk data.





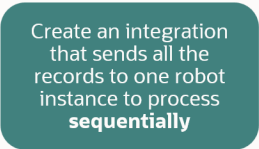
## Questions

Understand how the answers to the questions in the flow chart help inform your decision-making process.

| Question  | Example  | Why the question matters  |
|---|--|---|
|  <p>Do you need to process the records in order?</p> | <p>For example, do you need to update record 1, then record 2, and so on? Or can you update 5 records simultaneously?</p>          | <p>Robots can process orders in sequence without any issues. However, when your business requirements allow it, you'll find opportunities for efficiency by processing records in parallel.</p>   |
|  <p>Will all records process within 30 minutes?</p>  | <p>For instance, if you need to update 100 records, and each update takes 30 seconds, the total processing time is 50 minutes.</p> | <p>In general, when the total processing time for all records exceeds 30 minutes, Oracle recommends using an integration to manage the distribution of work across multiple robots. On the other hand, when the total processing time for all records is less than 30 minutes, you can allow the robot to manage the distribution of its own work and don't need a more robust solution architecture. The 30-minute time limit is an Oracle-recommended limit. Your organization can choose a different time period. Consider the amount of time you're willing to wait to determine whether a set of records processed successfully as well as the service limits. See Service Limits in <i>Provisioning and Administering Oracle Integration 3</i>.</p> |

## Processing Options

The flow chart provided several processing options. Review them in more detail.

| Processing option  | Description  | Use cases  |
|--|--|--|
|  <p>Create an integration that sends all the records to one robot instance to process <b>sequentially</b></p> | <p>Create an integration that passes the entire data set to a single robot instance. In the robot, create a foreach loop that iterates over all of the records, one at a time.</p> | <p>This solution is easy and straightforward and is best for records that can be processed relatively quickly and must be processed in a specific order.</p> |

| Processing option   | Description   | Use cases   |
|---|---|---|
| <p>Create an integration with a loop that sends records to one or more robot instances to process <b>sequentially</b></p> | <p>Create an integration with a foreach loop that handles sequential iterations. The integration iterates over all of the records, one at a time, and invokes a robot instance for each record in turn.</p> <p>For guidance on the number of robot instances to invoke and the number of records to pass to each robot, keep reading.</p>   | <p>This solution is ideal for the following scenarios:</p> <ul style="list-style-type: none"> <li>• Each record takes a long time to process.</li> <li>• You want to include error handling.</li> </ul> <p>If a single robot instance fails, you can add error handling to the integration so that the integration can continue sending records to other robot instances.</p> |
| <p>Create an integration with a loop that sends records to one or more robot instances to process <b>in parallel</b></p>  | <p>Create an integration with a foreach loop that handles parallel iterations. The integration processes the data in parallel. Each branch invokes a robot instance to process one or more records.</p> <p>For example, consider a data set with 100 records. An integration supports 5 parallel branches, and each branch calls 1 robot. Therefore, the integration and robot process 5 records at a time.</p> <p>For guidance on the number of robot instances to invoke and the number of records to pass to each robot, keep reading.</p> | <p>This solution is efficient when the total processing time for your records is high, either because you have a lot of records to process or because each record takes a long time to process, and your business requirements allow you to process the records in any order.</p>   |

## Additional Factors: Number of Robots and Records

Several scenarios require you to determine the number of robots that process records and the number of records that each robot processes.

The following scenarios require you to make these decisions:

Create an integration with a loop that sends records to one or more robot instances to process **sequentially**

Create an integration with a loop that sends records to one or more robot instances to process **in parallel**

Consider the following factors.

| Factor   | More information   |
|--|--|
| Overhead for calling a robot instance                      | <p>Each robot accomplishes one or more specific goals, such as updating a record. However, to achieve its goal, a robot must complete other tasks, such as opening an application, signing in, and navigating to the right page. All of the tasks that a robot does to prepare for its specific goal are the robot's overhead.</p> <p>For example, a robot that takes one minute and 15 seconds (1:15) to run might spend 1 minute navigating to the right page and then 15 seconds accomplishing its goal. That robot has 1 minute of overhead.</p>   |
| Total processing time for all records                      | <p>The following components determine the total processing time for all records:</p> <ul style="list-style-type: none"> <li>• Number of records to process</li> <li>• Overhead for the robot</li> <li>• Number of robots that process records</li> </ul> <p>For example, if you pass 3 records to a robot instance, you eliminate the overhead for 2 robots, but you also increase the total running time for the robot instance.</p>  |
| 30-minute (or a different organization-created) time limit | <p>The time limit is the amount of time that you're willing to wait before knowing whether a robot has succeeded. This value becomes the maximum processing time for a set of records and helps you calculate the number of records to send to a given robot instance. To maximize the efficiency of your automation, Oracle recommends passing the maximum number of records to the robot instance to limit the overhead time.</p> <p>Additionally, you can use parallel processing to reduce the clock time that passes before all records are processed. However, remember that each branch of the parallel processing incurs the overhead costs. Depending on the overhead duration and other components, distributing records to 5 branches might be less efficient than distributing records to only 3 branches.</p> |

## Sample Calculations

Sample calculations help you understand how to calculate the optimal number of robots to use and the number of records to send them.

## Simple Scenarios

A robot takes one minute and 15 seconds (1:15) to run. The robot spends 1 minute navigating to the right page and then 15 seconds accomplishing its goal. Different numbers of records and robot instances impact the total processing time for this work.

| Scenario   | More information   |
|--|--|
| Five robot instances each process one record, either sequentially or in parallel | <p>Each robot requires 1:15 to run, resulting in a total processing time of 6:25:<br/>1:15 processing time x 5 robots = 6:25 processing time</p> <p>The robots can run sequentially or in parallel.</p>  |
| One robot instance processes five records sequentially                           | <p>The robot requires 1 minute of overhead, and then 15 seconds of processing time for each record, resulting in a total processing time of 2:25:<br/>1 minute overhead + (15 seconds processing time per record x 5 records) = 2:15 processing time</p> |

| Scenario  | More information  |
|---|---|
| One robot instance processes 150 records sequentially | <p>Reducing overhead costs improves the efficiency of your automation, but passing too many records to a single robot instance can result in longer-than-preferred processing times.</p> <p>For instance, if one robot updates 150 records, you save 149 minutes of overhead time. However, the total processing time is 38:30, which might be longer than you want to wait to determine whether all the updates completed successfully.</p> <p><math>1 \text{ minute overhead} + (15 \text{ seconds processing time per record} \times 150 \text{ records}) = 38:30 \text{ processing time}</math></p> |

## Sequential Processing

If your business requires you to process records in sequence, determine the optimal number of tasks that each robot instance should process.

Here's how to complete these calculations.

### 1. Determine the overhead time

For example, consider a robot that spends 1 minute navigating to the right page and then 15 seconds accomplishing its goal. This robot has 1 minute, or 60 seconds, of overhead.

### 2. Determine the maximum time to process records

The 30-minute time limit contains 1,800 seconds:

$$30 \text{ minutes} \times 60 \text{ seconds} = 1,800 \text{ seconds}$$

Each record requires 60 seconds of overhead. You must subtract the overhead time from the maximum processing time:

$$1,800 \text{ seconds} - 60 \text{ seconds} = 1,740 \text{ seconds}$$

This calculation assumes that in 30 minutes, you complete the overhead one time and then use the rest of the time to process records.

### 3. Calculate the number of records that you can process

A robot needs 15 seconds to process each record.

To calculate the maximum number of records that a robot can process, divide the maximum time to process records by the time to process each record:

$$1,740 \text{ seconds maximum time} / 15 \text{ seconds per record} = 116 \text{ records}$$

**Theoretically, the optimal number of records for each robot to process is 116.**

#### Note:

This conclusion is theoretical because it makes several potentially faulty assumptions. For instance, the calculation assumes that the processing time never changes, but response times vary significantly in the real world. The optimal value according to a calculator doesn't reflect these varying circumstances. When making these decisions, consider building in some wiggle room that accommodates requirements that these calculations don't consider, such as network latency.

## Parallel Processing

If your business allows you to process records in parallel, you can distribute the work in a way that minimizes the time of the jobs.

### 1. Calculate the total potential overhead time

For example, if you have 100 records to process, and each record requires 60 seconds of overhead time, the total potential overhead is 6,000 seconds:

$$100 \text{ records} \times 60 \text{ seconds of overhead} = 6,000 \text{ seconds of potential overhead}$$

You can reduce this value by processing multiple records using a single robot instance.

### 2. Calculate the processing time without any overhead

For example, if each record requires 15 seconds to process (without its overhead time), the total processing time is 1,500 seconds:

$$15 \text{ seconds of processing time} \times 100 \text{ tasks} = 1,500 \text{ seconds}$$

You cannot reduce this time. However, you can reduce the amount of time that passes on the clock by processing records in parallel.

### 3. Consider several scenarios to find your preferred number of parallel branches (up to 5) and the number of records that each processes

To minimize the processing time, including overhead, you need to reduce your overhead time as much as possible while staying within the 30-minute time limit (or whatever time limit your organization chooses). Processing the records in parallel also minimizes the total time that passes on the clock before the jobs complete.

Calculate several scenarios to find your preferred combination. For example:

| Scenario   | Total processing time per branch  | Calculation   |
|--|---|---|
| 2 branches, 50 records per branch                              | 810 seconds (13 ½ minutes)  | 60 seconds of overhead + (50 records x 15 seconds of processing time) = 810 seconds       |
| 3 branches, 33 or 34 records per branch                        | 570 seconds (9 ½ minutes)   | 60 seconds of overhead + (34 records x 15 seconds of processing time) = 570 seconds       |
| 4 branches, 25 records per branch                              | 435 seconds (7 ¼ minutes)   | 60 seconds of overhead + (25 records x 15 seconds of processing time) = 435 seconds       |
| 5 branches, 20 records per branch                              | 360 seconds (6 minutes)   | 60 seconds of overhead + (20 records x 15 seconds of processing time) = 360 seconds       |
| 5 branches, 20 records per branch, sent in 2 different batches | Each batch finishes in 210 seconds (3 ½ minutes), for a total processing time of 420 seconds (7 minutes) per branch | [60 seconds of overhead + (10 records x 15 seconds of processing time)] x 2 = 420 seconds |

With a higher number of records or higher processing times, you might need to consider sending records to each branch in batches. This approach often reduces the processing time for a given batch of records but increases the total processing time. The following table provides sample calculations for processing 500 records in parallel.

| Scenario  | Total processing time per branch  | Calculation  |
|---|---|--|
| 2 branches, 250 records per branch                              | 3810 seconds (63 ½ minutes)<br>This value exceeds the 30-minute time limit  | 60 seconds of overhead + (250 records x 15 seconds of processing time) = 3810 seconds      |
| 3 branches, 166 or 167 records per branch                       | 2565 seconds (42 ¾ minutes)<br>This value exceeds the 30-minute time limit  | 60 seconds of overhead + (167 records x 15 seconds of processing time) = 2565 seconds      |
| 4 branches, 125 records per branch                              | 1935 seconds (32 ¼ minutes)<br>This value exceeds the 30-minute time limit  | 60 seconds of overhead + (125 records x 15 seconds of processing time) = 1935 seconds      |
| 4 branches, 125 records per branch, send in 2 different batches | Each batch finishes in 1005 seconds (16 ¾ minutes), for a total processing time of 2010 seconds (33 ½ minutes) per branch | [60 seconds of overhead + (63 records x 15 seconds of processing time)] x 2 = 2010 seconds |
| 5 branches, 100 records per branch                              | 1560 seconds (26 minutes)   | 60 seconds of overhead + (100 records x 15 seconds of processing time) = 1560 seconds      |
| 5 branches, 100 records per branch, sent in 2 different batches | Each batch finishes in 810 seconds (13 ½ minutes), for a total processing time of 1620 seconds (27 minutes) per branch    | [60 seconds of overhead + (50 records x 15 seconds of processing time)] x 2 = 1620 seconds |

#### 4. Choose the right scenario for your requirements

Consider your calculations. Build in some wiggle room for periods of higher-than-usual volume, network latency, and other unforeseen issues. Then, choose an approach.

Oracle recommends testing an integration and robot under load before going live to confirm that your approach will succeed in the real world.

## Guidelines for Expressions

When you define an action in a robot, you sometimes need to update a value, complete a calculation using the value, or perform an operation on the value. Complete these tasks by using expressions. For each expression, you must use the correct syntax.

### About Expressions

An expression performs an operation on a value and returns another value.

When building a robot, you can use expressions in many fields. The value that an expression works with can come from many places, including:

- An application that the robot interacts with.
- Another application.
- A variable, parameter, or property that you define.
- A value that you define.

For more real-world examples of expressions, see [Use Cases](#).

### Mathematical Operators

A mathematical operator performs math operations, such as addition and subtraction.

| Operator | Python equivalent | Description                                       | Expression | Returns |
|----------|-------------------|---|------------|---------|
| +        | +                 | Addition  | $\{2+2\}$  | 4       |
| -        | -                 | Subtraction                                       | $\{3-1\}$  | 2       |
| *        | *                 | Multiplication                                    | $\{3*2\}$  | 6       |
| /        | /                 | Division  | $\{8/4\}$  | 2       |
| %        | %                 | Modulo, which returns the remainder of a division | $\{3\%2\}$ | 1       |

### Comparison Operators

A comparison operator compares values and returns true or false.

| Operator | Python equivalent | Description              | Expression                   | Returns |
|----------|-------------------|--------------------------|------------------------------|---------|
| >        | >                 | Greater than             | $\{2>3\}$                    | false   |
| <        | <                 | Less than                | $\{1<3\}$                    | true    |
| >=       | >=                | Greater than or equal to | $\{2>=1\}$                   | false   |
| <=       | <=                | Less than or equal to    | $\{4<=5\}$                   | true    |
| ==       | ==                | Equal to                 | $\{1==1\}$<br>$\{one==one\}$ | true    |

## Logical Operators

A logical operator combines two comparisons and returns true or false.

| Operator | Python equivalent | Description                   | Expression                                | Returns |
|----------|-------------------|-------------------------------|---|---------|
| &&       | and               | Both statements must be true  | <code>{1&lt;=2 &amp;&amp; 2&lt;=3}</code> | true    |
|          | or                | Either statement must be true | <code>{2&lt;=1    3&lt;=2}</code>         | false   |

## Inversion Operators

An inversion operator returns the opposite result from what you'd expect.

| Operator | Python equivalent | Description   | Expression  |
|----------|-------------------|---|---|
| !        | not               | NOT operation, which inverts the result of a comparison<br>Use parentheses to clearly separate the NOT operation (!) from the expression. | <code>{!(1&gt;=2)}</code> returns true<br><code>{!(1&lt;=2 &amp;&amp; 2&lt;=3)}</code> returns false<br><code>{!(2&lt;=1    3&lt;=2)}</code> returns true |



## Functions

A function takes in data, processes it, and returns a result.

| Function | Python equivalent  | Description   | Format   | Example  |
|----------|--|---|--|--|
| toNumber | No direct equivalent, though you can use <code>int()</code> and <code>float()</code> to convert strings to numbers | Returns an integer or float for a numerical string. | $\$$<br><code>{toNumber (value )}</code><br>where:<br><i>value</i> is the number to return | <b>Return a number from a numeric string value</b><br><code>toNumber ("100.00")</code> returns 100.00<br><code>toNumber ("100,000")</code> returns 100000<br><code>toNumber ("100,000.00")</code> returns 100000.00<br><code>toNumber (\$VARIABLE.amount)</code> returns a value according to the value of the variable: <ul style="list-style-type: none"> <li>• If <code>\$VARIABLE.amount</code> is 100, returns 100</li> <li>• If <code>\$VARIABLE.amount</code> is 100.00, returns 100.00</li> </ul> If the string value contains any non-numeric characters, an error occurs. For example, <code>toNumber ("100A")</code> returns an error |

---

| Function | Python equivalent | Description                                 | Format  | Example  |
|----------|-------------------|---|---|--|
| toString | str()             | Returns a string that represents an object. | <pre>\$ {toString(value) }</pre> where:<br><i>value</i> is the string to return | <p><b>Return a string that represents a number</b></p> <pre>`\${toString(1)}`</pre> returns 1 <p><b>Return a person's first and last name</b></p> <p>You define a data type named <i>Person</i>. <i>Person</i> contains two properties: <i>First_name</i> and <i>Last_name</i>.</p> <p>Next, you define a variable, <i>var1</i>. Base <i>var1</i> on the <i>Person</i> data type.</p> <p>A robot returns the following values for the variable:</p> <ul style="list-style-type: none"><li><pre>var1.FirstName = "John"</pre></li><li><pre>var1.LastName = "Doe"</pre></li></ul> <p>Given this scenario, the following expression:</p> <pre>`\${toString(var1)}`</pre> <p>returns</p> <pre>{ "FirstName": "John",   "lastName": "Doe" }</pre> |

---

| Function  | Python equivalent               | Description   | Format   | Example   |
|-----------|---------------------------------|---|--|---|
| subString | <code>\$name[start:stop]</code> | Returns the specified characters within a string. Indexes are zero based. In plain English, a zero-based index means that you start counting the values at 0. For example, if you're using a substring function for the value ABC: <ul style="list-style-type: none"> <li>A is entry 0</li> <li>B is entry 1</li> <li>C is entry 2</li> </ul> | <pre>\$ {subString(string, startIndex, endIndex)}</pre> <p>where:</p> <ul style="list-style-type: none"> <li><i>string</i> is the string to evaluate</li> <li><i>startIndex</i> is the first character to include in the returned substring</li> <li><i>endIndex</i> is the last character to include in the returned substring</li> </ul> | <p><b>Return a subset of letters</b></p> <pre>\$ {subString(ABCD, 0, 0)} returns A  \$ {subString(ABCD, 1, 1)} returns B  \$ {subString(ABCD, 2, 4)} returns CDE</pre> <p><b>Return the last four digits of a phone number</b></p> <pre>\$ {subString(5551234567, 6, 9)} returns 4567</pre> |

## Formatting Rules for Expression Syntax





When you're creating robot actions or robot resources, you can enter expression syntax into many fields. You must use the correct syntax for each expression.

### General Formatting Rules

Some general rules apply to all expressions.

| Guideline  | More information   |
|--|--|
| Format all expressions using the following syntax: <code>{ }</code>                  | Enclose the entirety of every expression within curly brackets. For example:<br><pre>`\${toString(\$VARIABLE.CurrentInvoice[InvoiceAmount])} == \$VARIABLE.InvAmt &amp;&amp; \$VARIABLE.CurrentInvoice[SupplierName] == \$VARIABLE.SuppName`</pre> |
| Enclose all literal strings in quotation marks: <code>" "</code>                     | A literal string is any straight text that you type. For example:<br><pre>`{"https://www.mycompany.com/"}`</pre>   |
| You can use literal numbers but not exponents  | Supported values: <ul style="list-style-type: none"> <li>1, -1, and +1</li> <li>1.56, -1.56 and +1.56</li> </ul> Not supported values: <ul style="list-style-type: none"> <li>1e2</li> <li>2^2</li> </ul>  |
| Expect that the order of operations follows globally established rules of precedence | For example, the expressions on either side of a logical operator are evaluated before the logical operator is evaluated.  |



| Value   | Format  |
|---|---|
|  Output properties   | <p><b>Format for an output property</b></p> <pre>#{OUTPUT.output_property_name}</pre> <p>where:</p> <ul style="list-style-type: none"> <li><i>output_property_name</i> is the name of the output property.</li> </ul> <p><b>Format for a data type property on an output property</b></p> <p>Use this format when you base an output property on a custom data type with one or more properties, and you need to refer to one of the data type properties.</p> <ul style="list-style-type: none"> <li>Option 1: <pre>#{OUTPUT.output_property_name[property_name]}</pre><br/>You can include spaces in the <i>[property_name]</i> value.</li> <li>Option 2: <pre>#{OUTPUT.output_property_name.property_name}</pre><br/>No spaces are allowed in the <i>property_name</i> value, so you can use this option only if the data type's property has no spaces in its name.</li> </ul> <p>where:</p> <ul style="list-style-type: none"> <li><i>output_property_name</i> is the name of the property of the output.</li> <li><i>property_name</i> is the name of the data type property that the output property is based upon.</li> </ul> <p><b>Note:</b> Output properties can be nested. Follow the same formatting rules for these nested properties. For example:</p> <ul style="list-style-type: none"> <li><pre>#{OUTPUT.output_property_name[property_name] [property_name] [property_name]}</pre></li> <li><pre>#{OUTPUT.output_property_name.property_name.property_name.property_name}</pre></li> </ul> |
|  Page states       | <pre>#{pageState("page_state_name")}</pre> <p>where:</p> <ul style="list-style-type: none"> <li><i>page_state_name</i> is the name of the page state</li> </ul>   |
|  Robot connections | <pre>#{CONNECTION.robot_connection_name.property}</pre> <p>where:</p> <ul style="list-style-type: none"> <li><i>robot_connection_name</i> is the name of the robot connection</li> <li><i>property</i> is the name of the property in the robot connection</li> </ul>   |
|  Targets           | <pre>#{TARGET.target_name}</pre> <p>where:</p> <ul style="list-style-type: none"> <li><i>target_name</i> is the name of the target.</li> </ul>  |

| Value         | Format   |
|---------------|--|
| (x) Variables | <p><b>Format for a variable</b></p> <pre>#{VARIABLE.variable_name}</pre> <p>where:</p> <ul style="list-style-type: none"> <li><i>variable_name</i> is the name of the variable.</li> </ul> <p><b>Format for a data type property on a variable</b></p> <p>Use this format when you base a variable on a custom data type with one or more properties, and you need to refer to one of the properties.</p> <ul style="list-style-type: none"> <li>Option 1: <code>#{VARIABLE.variable_name[property_name]}</code><br/>You can include spaces in the <code>[property_name]</code> value.</li> <li>Option 2: <code>#{VARIABLE.variable_name.property_name}</code><br/>No spaces are allowed in the <code>property_name</code> value, so you can use this option only if the data type's property has no spaces in its name.</li> </ul> <p>where:</p> <ul style="list-style-type: none"> <li><i>variable_name</i> is the name of the variable.</li> <li><i>property_name</i> is the name of the property of the data type that the variable is based upon.</li> </ul> <p><b>Note:</b> Variables can have nested properties. Follow the same formatting rules for these nested properties. For example:</p> <ul style="list-style-type: none"> <li><code>#{VARIABLE.variable_name[property_name][property_name][property_name]}</code></li> <li><code>#{VARIABLE.variable_name.property_name.property_name.property_name}</code></li> </ul> |

## When to Convert a String to a Number

When you use the **get text** action to obtain a value from an application, Oracle Integration stores the value as a string. If you want to perform a numeric function on the value, such as adding it to another number or comparing it to another number, you must first convert the value to a number.

If you don't convert the value to a number, the numeric function won't work as expected.

Use the `toNumber` function to convert a string to a number. See [Functions](#).

# 4

## Build a Robot

Ready to build a robot? Familiarize yourself with key concepts in the quick starts, and then get started. Don't forget to complete the prerequisite tasks first.

### Topics:

- [Workflow for Building a Robot](#)
- [Quick Start for Building Robots](#)
- [Create Connections to Applications](#)
- [Create a Robot](#)
- [Add an Action to a Robot](#)
- [Add Logic to a Robot](#)
- [Specify Where a Robot Runs](#)
- [Activate a Robot](#)
- [Create and Update a Robot Resource](#)
- [Design an Integration That Calls a Robot](#)
- [View HTML and XPath](#)

## Complete Prerequisites

Before you start building robots, complete all the prerequisite tasks.

| Task                                  | Link(s)  |
|---------------------------------------|--|
| Review your network configuration     | 1. <a href="#">Review Your Network Configuration</a>   |
| Create accounts for robots and people | 2. <a href="#">Create Application Accounts</a>   |
| Create a confidential application     | 3. <a href="#">Create a Confidential Application</a> (Optional)<br>4. <a href="#">Ensure that the Confidential Application is Active</a> (Required only for troubleshooting)<br>5. <a href="#">Assign the ServiceDeployer Role to the Confidential Application</a> (Required only if you created a confidential application) |
| Install and configure the robot agent | 6. <a href="#">Meet the Robot Agent's Requirements</a><br>7. <a href="#">Download the Robot Agent</a><br>8. <a href="#">Update the Robot Agent's Configuration File</a><br>9. <a href="#">Start the Robot Agent</a><br>10. <a href="#">Start the Robot Agent Automatically</a> (Optional)                                    |
| Install the recorder                  | 11. <a href="#">Install the Recorder</a>   |
| Create a project                      | 12. <a href="#">Create a Project</a>   |

Next workflow: [Workflow for Planning a Robot](#).

## Review Your Network Configuration

The robot agent polls Oracle Integration for work. The robot agent must be able to contact Oracle Integration so that it can poll for work, and your network must allow this polling to occur.

- Review your organization's network configuration, and ensure that the robot agent can contact Oracle Integration so that it can poll for work.

For instance, ensure that your corporate firewall allows outbound calls to occur. If you need to update your allowlist, see *Obtain the Inbound and Outbound IP Addresses of the Oracle Integration Instance* in *Provisioning and Administering Oracle Integration 3*.

## Create Application Accounts

When a robot needs to sign in to an application to complete its work, the robot needs a user account. Additionally, whoever is building the robot requires an account with the same access and privileges.

### Accounts That You Might Need

You might need accounts for the robot and yourself in the application's test environment. Additionally, the robot probably requires an account for the production environment. The person building the robot should have the same access as the robot.

#### Create Accounts:

1. Identify all of the required accounts, including their access level, in the applications that you're automating.
2. Work with an administrator to create all of the required accounts.

## Create a Confidential Application

When you provision your Oracle Integration instance, Oracle creates a default confidential application for all of your robot agents to use. If you want some robot agents to use different credentials, you can create additional confidential applications.

### What Is a Confidential Application?

A confidential application is an OAuth client application that allows robot agents to securely connect to Oracle Integration using the OAuth protocol. Organizations typically create additional confidential to increase security. For example, consider an organization that builds robots for human resources and financial applications. All of the HR robot agents can use one confidential application, and the financial robot agents can use another confidential application.

#### Create a Confidential Application:

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Navigate to the Integrated applications page.
  - a. Open the navigation menu and select **Identity & Security**. Under **Identity**, select **Domains**.

The Domains page is displayed.
  - b. If not already selected, select the **Compartment** that holds the domain in which you want to create the confidential application.



- c. In the **Name** column, select the domain in which you want to create the confidential application.  
  
You must work in the domain in which your organization created the Oracle Integration instance.  
  
The Overview page for the domain is displayed.
  - d. In the left menu below Identity domain, select **Integrated applications**.
3. Add a confidential application.
    - a. Select **Add application**.  
  
The Add application wizard appears.
    - b. Select **Confidential Application**, and select **Launch workflow**.
    - c. Enter a name and description for the confidential application, and select **Next**.  
  
You don't need to fill in any other fields on this page.
    - d. On the Configure OAuth tab, fill in the following fields:
      - Select **Configure this application as a client now**.
      - Below Authorization, select only **Client credentials**.
      - Below Token issuance policy, select **Specific** and **Add resources**.
      - Below Resources, select **Add scope**, select the name of the Oracle Integration instance that the confidential application is associated with, and select **Add**.
    - e. Select **Next**, and then **Finish**.

## Ensure that the Confidential Application is Active

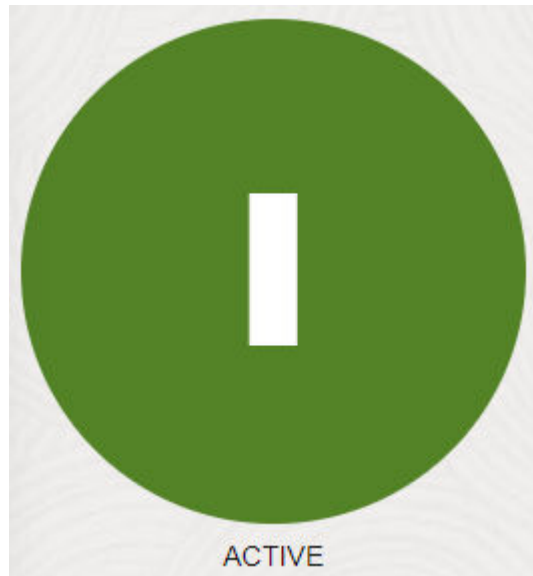
The confidential application that the robot agent uses must be active, or the robot agent can't start running. You typically need to check whether the confidential application is active only if you experience issues with the robot agent.

1. Sign in to the Oracle Cloud Infrastructure Console.
2. Navigate to the Integrated applications page.
  - a. Open the navigation menu and select **Identity & Security**. Under **Identity**, select **Domains**.  
  
The Domains page is displayed.
  - b. If not already selected, select the **Compartment** that holds the domain in which you want to create the confidential application.
  - c. In the **Name** column, select the domain that holds the confidential application.  
  
The Overview page for the domain is displayed.
  - d. In the left menu below Identity domain, select **Integrated applications**.
3. Verify that the confidential application is active.
  - a. Search for the confidential application that the robot agent uses.  
  
The name of the confidential application that Oracle created for you ends with **RPA\_AGENT\_DEFAULT\_APP**, and its description is **Integration Cloud Services RPA Agent**.

 **Note:**

If you don't see an Oracle-created confidential application, you can enter a service request (SR), and Oracle can create the application for you.

- b. In the Name column, select the confidential application.
- c. To the left of the name, check the status.  
If Active appears, the confidential application is active.



- d. If the confidential application isn't active, investigate the reason.  
For instance, if your organization experienced a security issue, someone might have deactivated the confidential application and created a new confidential application.

## Assign the ServiceDeployer Role to the Confidential Application

A confidential application requires the ServiceDeployer role so that the robot agent can access the Oracle Integration APIs. When Oracle creates a default confidential application for you, Oracle assigns this role to the application. Therefore, you must complete this step only if you create an additional confidential application.

1. Open a confidential application.
  - a. Sign in to the Oracle Cloud Infrastructure Console.
  - b. Open the navigation menu and select **Identity & Security**. Under **Identity**, select **Domains**.  
The Domains page is displayed.
  - c. If not already selected, select the **Compartment** that holds the domain in which you want to create the confidential application.
  - d. In the **Name** column, select the domain in which you want to create the confidential application.  
The Overview page for the domain is displayed.
  - e. In the left menu below Identity domain, select **Oracle Cloud Services**.

- f. Search for the confidential application that you just created.  
The name of the confidential application that Oracle created for you ends with **RPA\_AGENT\_DEFAULT\_APP**, and its description is **Integration Cloud Services RPA Agent**.
  - g. In the Name column, select the confidential application to update.
2. Assign the ServiceDeployer role to the confidential application.
    - a. In the left menu, below Resources, select **Application roles**.
    - b. In the Application roles table, find the **ServiceDeployer** role, and at the end of its row, click the arrow to expand the entry.
    - c. Next to Assigned applications, select **Manage**.
    - d. Find the confidential application in the list and select it.
    - e. Select **Close**.

## Meet the Robot Agent's Requirements

The computer where a robot runs, typically a Windows virtual machine (VM), is called an environment. You must ensure that all computers meet the system requirements for the robot agent.

- Ensure that the computers meet the system requirements for the robot agent.  
See [System Requirements](#).

## Download the Robot Agent

Download and install the robot agent on every virtual machine (VM) or computer that runs a robot.

### Learn More About the Robot Agent

To learn more about the robot agent, see [About the Robot Agent](#) and [Specifications](#).

### Where You Typically Download the Robot Agent

You typically download the robot agent on the following computers:

- The work computers of the people who build robots  
If robot builders install the robot agent on their computers, they can test their robots as they build them.
- The environments that the robots will run on for formal testing
- The environments that the robots will run on in production

### Download the Robot Agent:

1. On the computer where you need to install the robot agent, download the ZIP file for the robot agent  
The ZIP file contains a single folder with all the files that you need. One of the files is a readme that contains a link to this page.
  - a. Sign in to the Oracle Integration instance that the robot agent needs to connect to.
  - b. In the navigation pane, click **Design**, then **Agents**.

- c. Select **Download**, then **Robot Agent**.

The ZIP file is downloaded to the computer, often to the Downloads folder. The download typically takes a minute or less, though sometimes it takes a few moments for the download to begin.

2. Extract the ZIP file to the directory of your choice on the computer.

 **WARNING:**

Do not include any spaces in the folder name, or you won't be able to install the robot agent. Additionally, Oracle recommends not including any spaces in the directory path.

The files need to remain on the computer for as long as the robot agent runs on the computer, so choose a location carefully. For instance, don't leave the files in your Downloads folder, in case you or someone else inadvertently deletes the files in the future.

## Update the Robot Agent's Configuration File

Every robot agent contains a configuration file, which contains properties for the robot agent's name, secrets, and service instance. For every robot agent, you must enter the robot agent's name. Additionally, if the robot agent communicates with a confidential application, you must update several other properties.

1. Get the client secret and scope values from the confidential application.

These steps are required only if the robot agent communicates with a confidential application that your organization created. **If you're using the default confidential application that Oracle created, skip to the next step.**

- a. Sign in to the Oracle Cloud Infrastructure Console.
- b. Open the navigation menu and select **Identity & Security**. Under **Identity**, select **Domains**.

The Domains page is displayed.

- c. If not already selected, select the **Compartment** that holds the domain with the confidential application.

- d. In the **Name** column, select the domain that holds the confidential application.

The Overview page for the domain is displayed.

- e. In the left menu below Identity domain, select **Integrated applications**.

- f. In the search box above the table, search for the confidential application.

The name of the confidential application that Oracle created for you ends with `RPA_AGENT_DEFAULT_APP`, and its description is **Integration Cloud Services RPA Agent**.

- g. Scroll down, and find the client secret and scope values.

These values are sensitive, so be mindful of where you place them after copying them.

| Value you need | Where to find it   |
|----------------|--|
| Client ID      | Below the <b>General Information</b> heading, find the <b>Client ID</b> entry, and copy the value. |

| Value you need | Where to find it  |
|----------------|---|
| Client secret  | Below the <b>General Information</b> heading, find the <b>Client secret</b> entry. Select <b>Show secret</b> , and copy the value.  |
| Scope          | Below the <b>Token issuance</b> policy heading, and then below the <b>Resources</b> heading, review the entries in the table.<br>If multiple entries appear, copy the value that ends with <code>consumer::all</code> . |

- On the computer where you installed the robot agent, open the folder that contains the agent and its related files.
- Open the `InstallerProfile.cfg` file in a text editor, such as Notepad.

The file contains the following properties. The `DISPLAY_NAME=` property is blank, but the other properties have values specified.

```
DISPLAY_NAME=
IDCS_CLIENT_ID=
IDCS_CLIENT_SECRET=
IDCS_SCOPE=
IDCS_URL=
SERVICE_INSTANCE_ENDPOINT=
SERVICE_INSTANCE_ID=
```

- Update the properties in the file as needed.

| Property   | Value to enter   |
|--|--|
| <code>DISPLAY_NAME=</code>   | <b>If you're using the default confidential application that Oracle created, this is the only property that you need to define.</b><br>Enter the name of the robot agent in Oracle Integration.<br>Choose a value carefully, and consider creating naming conventions for your robots. For example, you might want to distinguish individuals' local computers from the virtual machines (VMs) that host robots, and you might want to distinguish VMs for each robot.   |
| <code>IDCS_CLIENT_ID=</code><br><code>IDCS_CLIENT_SECRET=</code><br><code>IDCS_SCOPE=</code>           | If the robot agent uses a confidential application that your organization created, update the values so that they reference the confidential application: <ul style="list-style-type: none"> <li><code>IDCS_CLIENT_ID=</code>: Client ID of the confidential application that the robot agent uses.</li> <li><code>IDCS_CLIENT_SECRET=</code>: Client secret of the confidential application that the robot agent uses.</li> <li><code>IDCS_SCOPE=</code>: Scope of the confidential application that the robot agent uses.</li> </ul> A previous step in this procedure describes how to find these values. |
| <code>IDCS_URL=</code><br><code>SERVICE_INSTANCE_ENDPOINT=</code><br><code>SERVICE_INSTANCE_ID=</code> | Do not update the values of any of these properties, even if the robot agent uses a confidential application that your organization created.   |

- Save and close the file.
- Repeat these steps as needed for other robot agents that you installed.

## Start the Robot Agent

You must start the robot agent from the computer on which it is installed. If a robot agent isn't running on a computer, the computer can't run any robots.

1. Open the location to which you extracted the ZIP file for the robot agent.
2. Note the version number in the `orpa-agent` JAR file.

For instance, for the `orpa-agent-0.1.80.jar` file, the version number is `0.1.80`.

3. Open a command-line tool.

You have the following options.

| Operating system | Places where you can start the robot agent   |
|------------------|--|
| Windows          | <ul style="list-style-type: none"> <li>• <b>Command Prompt</b><br/>Open a Command Prompt, and navigate to the location of the unzipped files.</li> <li>• <b>Command Prompt as an administrator</b><br/>If your Windows machine doesn't allow you to run scripts from the command line, you can start the robot agent from a Command Prompt that you run as an administrator.<br/><br/>Run a Command Prompt as an administrator, and navigate to the location of the unzipped files.</li> </ul> |
| MacOS            | Open the <b>Terminal</b> app, and navigate to the location of the unzipped files.  |

4. Build the following command:

```
java -jar agent_install_directory-folder_name\orpa-agent-
name_of_agent_JAR_file
```

where:

- `agent_install_location` is the location of the unzipped files. Oracle recommends not including any spaces in the directory path.
- `folder_name` is the name of the folder that contains the robot agent's files. Do not include any spaces in the folder name, or you won't be able to install the robot agent.
- `name_of_agent_JAR_file` is the name of the robot agent's JAR file, including its version number in the `X.X.XX` format (such as `0.1.80`) and its `.jar` extension.

For example, if the folder is on your `C:\` drive, you named the folder `robot_agent`, and the JAR file is named `orpa-agent-0.1.80.jar`, the command looks like this:

```
java -jar C:\robot_agent\orpa-agent-0.1.80.jar
```

5. Run the command in the command-line tool.

Information messages appear. When the following message appears, the robot agent has started successfully:

```
INFO - Requesting messages from ControlRoom
```

After the robot agent starts successfully for the first time, Oracle Integration removes the values of the `IDCS_CLIENT_ID=` and `IDCS_CLIENT_SECRET=` from the file and writes them to the computer's local keystore for security properties. For auditing purposes, Oracle

Integration adds a commented-out message to the file about the change. During subsequent starts of the robot agent, the values are read from the local keystore.

6. If you don't see the previous message, or if any errors appear after you try to start the agent, run the command again, up to a few times if required.

Network issues sometimes prevent the robot agent from starting.

**What if the robot agent doesn't start successfully?** Troubleshoot the issue. See [Cannot Start a Robot Agent](#).

If you shut down the computer that runs the robot agent, the robot agent stops running. You must start the agent again after you restart the computer, or configure the robot agent to start automatically. See [Start the Robot Agent Automatically](#).

## Start the Robot Agent Automatically

If you want the robot agent to automatically start and stop with the Windows operating system, configure the robot agent as a Windows service. This task is optional and applies only if the robot agent runs on a computer or virtual machine with a Windows operating system.

### **Caution:**

If you don't configure the robot agent as a Windows service, you must manually start the robot agent every time the VM restarts.

You use an open-source application called NSSM to configure the robot agent as a Windows service.

1. Download NSSM to the computer.
  - a. Download NSSM from the [NSSM website](#).

Download the file to the computer that hosts the robot agent, or to a location that you can access from that computer.
  - b. Unzip the NSSM file.
  - c. Place the NSSM files in a location of your choice on the computer.
2. Locate the nssm.exe file in the NSSM files, and copy the path to its location.
3. If the robot agent isn't already running, start the robot agent.

See [Start the Robot Agent](#).
4. Add the path for the NSSM service to your Windows environment variables.
  - a. Open the Settings dialog in Windows.

For example, open the Windows menu, and type `settings`.
  - b. Next to **Related links**, select **Advanced system settings**.

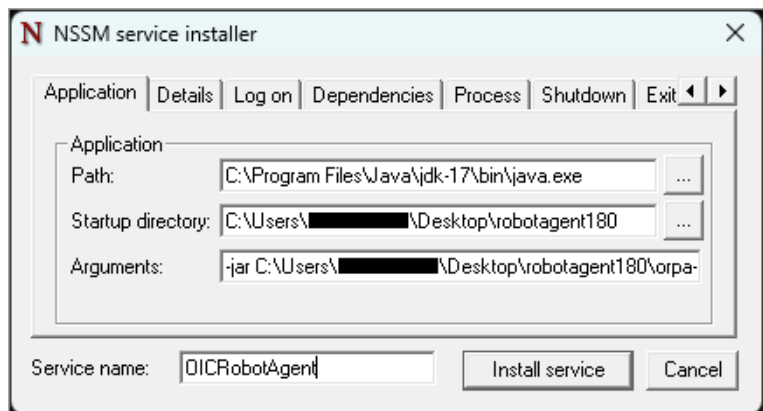
The System Properties dialog opens.
  - c. Select **Environment Variables**.

The Environment Variables dialog opens.
  - d. In the **User variables** list, select the `Path` variable, and select **Edit**.

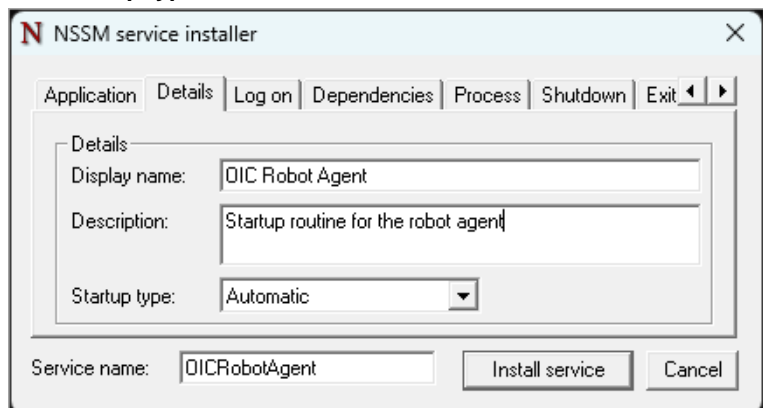
The Edit environment variable dialog opens.

- e. Click **New**.  
A new row becomes editable in the Edit environment variable dialog.
  - f. Paste the path that you copied in a previous step into the row.  
The path shouldn't include `nssm.exe` file name.
  - g. Click **OK**, and then click **OK** again.
5. Update the
- a. Open a command prompt as an administrator.
  - b. Enter the following command: `nssm install`  
The NSSM service installer dialog box appears.
  - c. Fill in the fields.

| Tab         | Fields to enter  |
|-------------|--|
| Application | <ul style="list-style-type: none"> <li>• <b>Path:</b> Enter the directory of the <code>java.exe</code> file, including the <code>java.exe</code> file name.</li> <li>• <b>Startup directory:</b> Enter the directory of the robot agent.</li> <li>• <b>Arguments:</b> Enter the command that starts the robot agent, without <b>java</b> at the beginning. If you're not sure, see <a href="#">Blue the Robot Agent</a>.</li> <li>• <b>Service name:</b> Enter the name of the service as you want it to appear in the list of Windows services. Do not include spaces in the name.</li> </ul> |

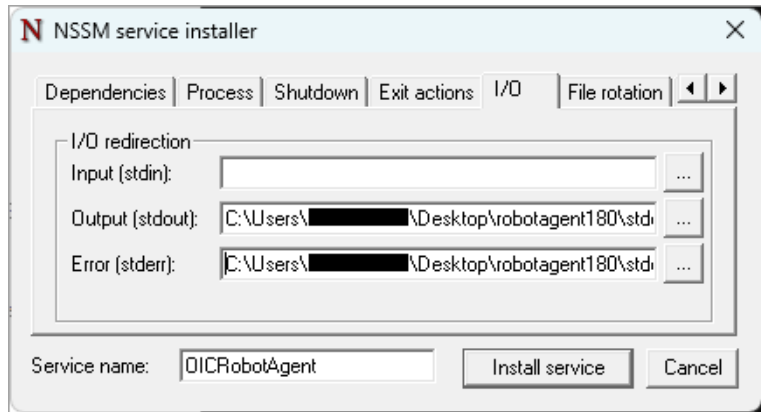


- |         |   |
|---------|---|
| Details | <ul style="list-style-type: none"> <li>• <b>Display name:</b> Enter the display name for the service.</li> <li>• <b>Description:</b> Enter a description for the service.</li> <li>• <b>Startup type:</b> Select <b>Automatic</b>.</li> </ul> |
|---------|---|

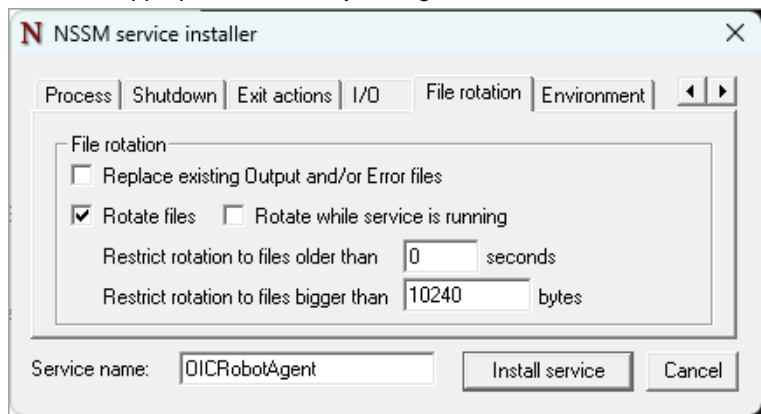




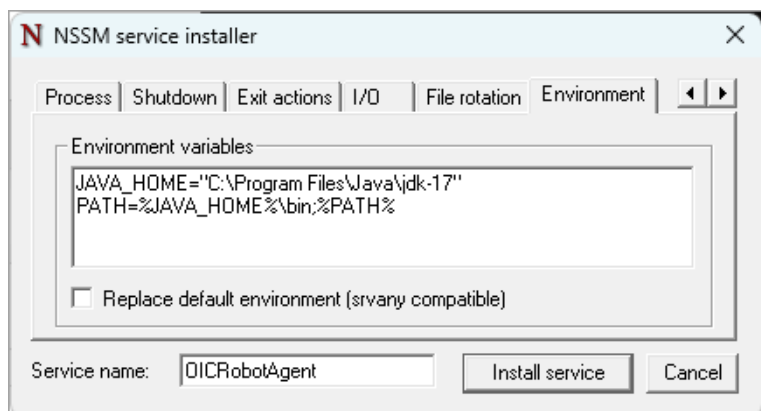
| Tab | Fields to enter   |
|-----|---|
| I/O | <ul style="list-style-type: none"> <li>• <b>Output (stdout):</b> Select the location for the output files for the service, including the name of the output file.</li> <li>• <b>Error (stderr):</b> Select the location for the error files for the service, including the name of the error file.</li> </ul> |



|               |   |
|---------------|---|
| File rotation | <ul style="list-style-type: none"> <li>• Select only <b>Rotate files</b>.</li> <li>• For <b>Restrict rotation to files bigger than ___ bytes</b>, enter 10240 or an appropriate value for your organization.</li> </ul> |
|---------------|---|



|             |   |
|-------------|---|
| Environment | <p><b>Environment variables:</b> Enter the location of the Java home, such as:<br/>           JAVA_HOME="C:\Program Files\Java\jdk-17"<br/>           PATH=%JAVA_HOME%\bin;%PATH%</p> |
|-------------|---|



- d. Click **Install service**.

In the command prompt, a message informs you that the service was installed successfully.

6. Start the service.
  - a. Open the Services dialog in Windows.

For example, open the Windows menu, and type `settings`.
  - b. Find the service you just created. Its name is the Display name that you specified.
  - c. Start the service.
7. To verify that the service started as expected, check the output and error files for any errors.
8. Repeat the previous steps as needed for other Windows computers or virtual machines for which the robot agent must start automatically.


## Install the Recorder


The recorder is a Google Chrome browser extension that you use to build robots. Everyone who builds robots must install the recorder.


1. Open Oracle Integration in the Google Chrome browser.
2. Open any robot.

### Tip:


If you haven't created a robot yet, you can create one now. See [Create a Robot](#).

- a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.
3. Navigate to the download area for the recorder extension.
    - a. In the toolbar, select **Targets** .



The Targets panel appears.
  - b. Select **Create** .

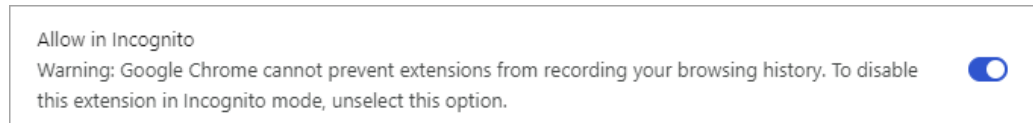
The Target panel appears.

  - c. Click within the **Locator** field, and select **Target a page element** .

The Browser extension not detected message appears.
4. Download the recorder extension: Below the Browser extension not detected message, select **Download browser extension**.

A ZIP file downloads locally, typically in your Downloads folder.
5. Install the recorder.

- a. Navigate to the location where the ZIP file downloaded, such as your **Downloads** folder.
  - b. Extract the ZIP file to a location of your choice.
  - c. In Google Chrome, open the **Extensions**  menu, and select **Manage Extensions**.
  - d. In the upper-right corner, select **Developer mode**, if it's not already selected.  
Don't skip this step, or you won't be able to install the extension.
  - e. On the Google Chrome Extensions page, select **Load unpacked**, located below the page title.
  - f. Navigate to the folder where you unzipped the file, select the `target` folder, and select **Select Folder**.
6. Close and reopen the Google Chrome browser.
  7. If you work in an Incognito window, allow the extension in Incognito windows.  
You can perform these steps in an Incognito browser or a regular Chrome browser.
    - a. In Google Chrome, open the **Extensions**  menu, and select **Manage Extensions**.
    - b. In the box for the Oracle Robot Designer Extension, select **Details**.
    - c. Enable the **Allow in Incognito** setting.



## Create a Project

Build a robot within a project. If you don't have an existing project to work in, create a new project.

To learn about projects, including how to keep them organized, see *Get Started with Projects in Using Integrations in Oracle Integration 3*.

- Create a project.  
See *Create or Import a Project in Using Integrations in Oracle Integration 3*.

## Quick Start for Building Robots

Get started building quickly, confidently, and correctly with videos, tips, and more.

### Topics:

- [Workflow for Planning a Robot](#)
- [Workflow for Building a Robot](#)
- [Learn with a Tutorial](#)
- [11 Tips for New Robot Builders](#)
- [Tailor Your Building Experience](#)

## Workflow for Planning a Robot

Don't skip the critical step of planning and designing your automation. Spending time planning saves you a lot of time reworking and helps ensure that your automation achieves your business goals.

Previous workflow: [Complete Prerequisites](#).

### Identify the Problem

Your first step in planning a robot is familiarizing yourself with the business process that it will address, including the problems that the business process solves.

| Task  | More information   |
|---|--|
| Identify a business process that is impacting your business           | <p>How you define a business impact is up to you. For instance, the impact could be one or more of the following:</p> <ul style="list-style-type: none"> <li>• Inefficiencies</li> <li>• Reduced effectiveness</li> <li>• Limited or no oversight and observability</li> <li>• Lack of continuous improvement</li> </ul>   |
| Identify the stakeholders   | <p>Identify the stakeholders for the business process, including:</p> <ul style="list-style-type: none"> <li>• The people who complete any steps in the current process and anyone in the organization who has an interest in the process.</li> <li>• Anyone who is a stakeholder for the automation work.</li> </ul>  |
| Review the current business process and the problem that it addresses | <p>Gather information about the business process that is impacting your organization. If possible, meet with the stakeholders to collect information about the current state of the business process and the problem that the process addresses.</p> <p>Focus on the current situation rather than on opportunities for improvement or on implementation details.</p> <p>The deep knowledge that you gain will help you plan the right solution.</p> |

| Task   | More information   |
|--|--|
| Assess the need to redesign the business process | <p>Determine whether the process is working for your organization:</p> <ul style="list-style-type: none"> <li> <b>Process that isn't working</b><br/>           Some business processes are inherently flawed. For example, if your current process generates invalid orders, automation doesn't make sense. Generating invalid orders more efficiently is not a win. Instead, redesign the process before automating it.         </li> <li> <b>Process that is working</b><br/>           Some business processes are perfect, but most have room for improvement. How do you know whether to rethink the process or automate it as-is? After all, automating a bad process is generally a bad idea.<br/><br/>           Here's what Oracle recommends: If the business process that you're currently using is working, you can benefit from automating it <i>today</i>. The automation doesn't change or improve the business process, but you gain immediate value from the automation. For instance:           <ul style="list-style-type: none"> <li>The business process is now automated, with potential for immediate gains in efficiency and effectiveness.</li> <li>You gain insight into how the business process is working, so you can start identifying the areas to improve.</li> </ul> </li> </ul> <p><b>Note:</b> Automating an inefficient process might seem unproductive. But, consider the cost of rethinking a business process. You could spend years assessing and evaluating the process and reaching consensus with all stakeholders. That's years of working without automation. Instead, Oracle recommends automating today, and making tactical improvements in the future using the insight that you gain.</p> |

## Understand the Applications

Familiarize yourself with the applications that you're automating and their APIs, if applicable.

| Task  | More information   |
|---|--|
| Understand the applications                                   | Familiarize yourself with the applications that you're automating. For instance, while interviewing users, you might ask for demos of the business process that you'll automate.   |
| Determine whether the applications have APIs                  | <p>Determine whether an application has APIs so you can understand your options for automation:</p> <ul style="list-style-type: none"> <li>If an application doesn't have APIs, you can still automate it using robots.</li> <li>If an application has APIs, you can automate it using integrations, too.</li> </ul> |
| If the applications have APIs, familiarize yourself with them | <p>Familiarize yourself with the application's APIs by reviewing their documentation. This knowledge helps you determine the right automation for a given task.</p> <p>For example, if APIs are available for the task that you want to automate, you can consider designing an integration for the automation.</p>  |

## Define the Requirements

After collecting information about the use cases for your automation, define and prioritize your requirements.

| Task                                   | More information  |
|--|---|
| Identify your requirements             | Using the use cases that you collected, write the requirements for your automation solution.<br><br>Be as specific as you prefer, and work in the format that your team usually uses. For instance, you might write a product requirement document, or you might create one or more tickets in your issue tracking software.  |
| If needed, prioritize the requirements | Determine the timeline for work, the scope of the work, and the availability of resources.<br><br>Next, determine whether you can deliver all requirements in the initial delivery of the automation.<br><br>If you need to split the work into multiple sprints, phases, or releases, work with your stakeholders to prioritize the requirements.  |
| Plan your implementation and work      | Determine how you will implement your requirements. For instance: <ul style="list-style-type: none"> <li>Identify the number of integrations and robots to design and build. See <a href="#">When to Create Robots Versus Integrations</a>.</li> <li>Specify the actions that the integrations and robots will take.</li> <li>Determine how many environments you must set up, and create system requirements for them.</li> <li>Create a schedule for the work.</li> </ul> |

Next workflow: [Workflow for Building a Robot](#).

## Workflow for Building a Robot

To build a robot, provide information about the application(s) that the robot works in and define the activities that the robot completes. You also need to design the integration that calls the robot.

Previous workflow: [Workflow for Planning a Robot](#).

| Step | Task  | More information  |
|------|---|---|
| 1    | <a href="#">Complete Prerequisites</a>                    | For example, create accounts, and install and configure the robot agent.  |
| 2    | <a href="#">Create a Robot Connection Type (Optional)</a> | A <a href="#">robot connection type</a> specifies the parameters that you use to connect to an application. For example, a web application might require a user name, password, and URL. If the <a href="#">predefined robot connection types</a> don't meet your needs, create a new one.  |
| 3    | <a href="#">Create a Robot Connection</a>                 | Base the robot connection on a robot connection type.   |
| 4    | <a href="#">Create a Robot</a>                            | After creating a robot, define the steps that the robot completes by adding robot actions and logic to the robot: <ul style="list-style-type: none"> <li><a href="#">Add an Action to a Robot</a></li> <li><a href="#">Add Logic to a Robot</a></li> </ul> Before or while creating the robot, add the robot resources that the robot uses, such as a trigger or variable. See <a href="#">Create and Update a Robot Resource</a> . |
| 5    | <a href="#">Specify Where a Robot Runs</a>                | Create an environment pool, add computers to it, and associate your robot with the environment pool.  |

| Step | Task   | More information   |
|------|--|--|
| 6    | <a href="#">Design an Integration That Calls a Robot</a> | Robot and integration developers can work concurrently or at different times. A robot developer must complete a few quick tasks before an integration developer can start their work.<br><br>Complete this task at any time. |

Next workflow: [Workflow for Testing a Robot](#).

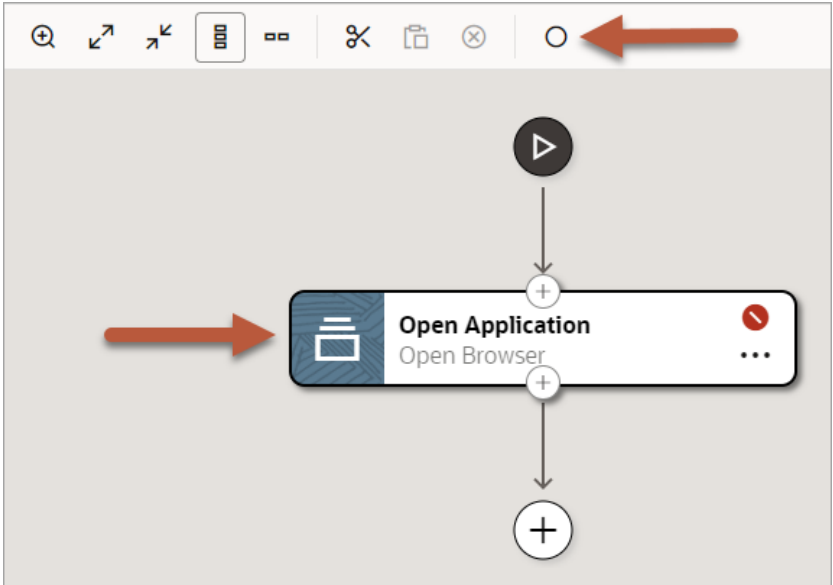
## Learn with a Tutorial

Ready to build your first robot? Walk through the process by following a step-by-step tutorial.

| Tutorial  | Description   |
|---|---|
| <a href="#">Get Started with Robotic Process Automation (RPA)</a> | Learn how to build a robot and the integration that calls it. |

## 11 Tips for New Robot Builders

While building a robot, you'll record the actions that a robot completes using the recorder, all while harnessing the power of control and power of the low-code capabilities. Keep reading to learn useful tips and tricks for these building tools.

| Tip   | More information  |
|---|---|
| 1. <b>Select an action, and then start the recorder</b> | <p>To enable the <b>Record after the selected action</b> <input type="radio"/> button, select an action.</p>  <p>The screenshot shows a workflow editor with a toolbar at the top. The toolbar contains several icons: a magnifying glass, a double arrow, a left arrow, a list icon, a play button, a scissors icon, a folder icon, a close icon, and a radio button. A red arrow points to the radio button. Below the toolbar is a workflow diagram with a central action box labeled 'Open Application' with the sub-action 'Open Browser'. A red arrow points to this action box. The diagram shows a play button at the top, followed by a plus sign, then the 'Open Application' box, then another plus sign, and finally a plus sign at the bottom.</p> |

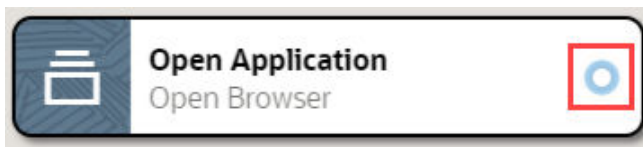
| Tip | More information |
|-----|------------------|
|-----|------------------|

**2. Understand where the recorder adds actions**

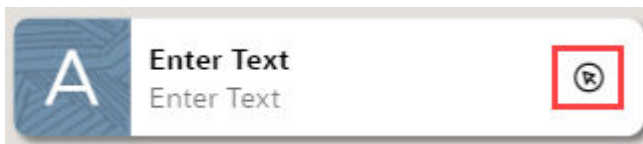
When you select an action on the canvas and then select **Record**, the recorder adds the new actions *after* the currently selected action, with the following exceptions:

- **Foreach loop**  
When you record from a foreach loop, the recorder asks whether you want to add the action after or within the foreach loop.
- **Switch condition**  
When you record from a switch condition, the recorder always adds the actions within the condition, since you cannot record after a condition branch.

**Bonus tip:** If you missed adding an action, you can still add it, even if the recorder is still running. Note that the **Recording cursor location** icon appears on the currently selected action.



To select a different action, simply point to another action, and select **Place recording cursor**.



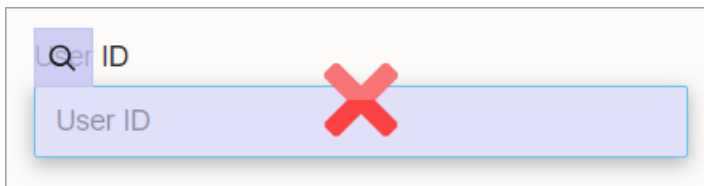
The next action that you record is inserted after the selected action.

**3. Open an application, and then start the recorder**

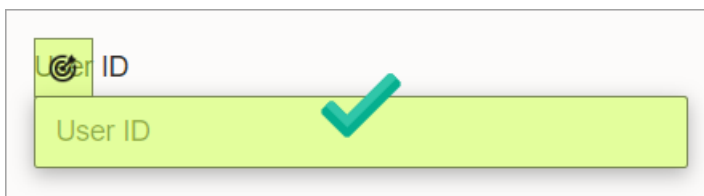
You can record actions only in the browsers that are open before you start the recorder. But, not to worry: If you open an application after starting the recorder, all you need to do is stop the recorder, and then start it again.

**4. Wait for the green shading**

When recording or when targeting a field, hover your cursor over a UI element and note the shading that appears. While the shading is purple and the icon above the field is a magnifying glass, the recorder is still collecting information about the element. **Don't select the element yet.**




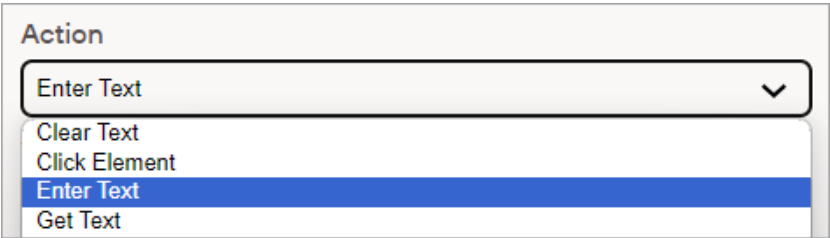
After the shading turns green and the icon changes to a target, select the UI element.








**5. If the whole page turns green, press Esc**

If you hover over an element that covers the entire page, and the entire page becomes shaded green, cancel the targeting by pressing **Esc**.




| Tip  | More information   |
|--|--|
| <b>6. Select the correct UI element</b>                          | <p>When you point to a UI element while recording, you can often target multiple options, such as the icon on a button, the button itself, and an element that encloses the button. Most of the time, you can target any of these options, and the robot works as expected.</p> <p>Additionally, if the robot must enter text into a field, select the field, not its label.</p>   |
| <b>7. Select the correct robot action</b>                        | <p>While you select a field while recording, a drop-down list shows all of the available actions for the UI control.</p>   |
| <b>8. Familiarize yourself with an application's HTML</b>        | <p>To build a robot using the low-code capabilities, you should have some familiarity with reviewing the HTML of a web page.</p> <p>For example, you should understand the structure of the website or application that you're interacting with and feel comfortable viewing the source for the application's HTML.</p> <p>This familiarity is helpful because the appearance of a UI element is sometimes different from its underlying HTML code. For instance, if a robot must interact with a checkbox, you might add the checkbox action to the robot. However, if the checkbox isn't coded as a checkbox in its HTML, the checkbox action can't interact with the element.</p> <p>And remember: If you don't want to wade through a page's HTML, use the recorder to build your robot. The robot reviews the HTML for each element that you select and provides the available actions for the element.</p> <p>For help working with HTML and XPath, see <a href="#">View HTML and XPath</a>.</p> |
| <b>9. Customize Oracle Integration to fit your working style</b> | <p>Some people like to define the big picture and then provide the details, and others prefer the opposite. Still others prefer to build everything as they go. Oracle Integration supports all of these working styles. For example:</p> <ul style="list-style-type: none"><li>• Optimize your working experience to match your preferences by updating your build settings.<br/>See <a href="#">Tailor Your Building Experience</a>.</li><li>• Understand your options for building robot resources.<br/>Every robot requires a trigger. Additionally, most robots require variables, validation, and custom data types. The timing of when you create these resources is up to you: Either before or as you build the robot.<br/>See <a href="#">Create and Update a Robot Resource</a>.</li></ul>  |


| Tip   | More information   |
|---|--|
| 10. To find stuff fast, use search                  | <p>When you have a robot with many actions, finding the action you need to update can take a little time. A Search panel helps you find what you need quickly.</p> <ul style="list-style-type: none"> <li>In the toolbar to the right of the canvas, click <b>Search</b> . The Search panel contains a tree that lists all the actions in the robot, including their names and unique identifiers, prefaced with <code>lc</code> (<code>lc1</code>, <code>lc2</code>, and so on).</li> </ul>  |
| 11. Choose a layout that suits you and your monitor | <p>If your monitor looks like it belongs in a movie theater, you might prefer to see your robot in a horizontal layout.</p> <ul style="list-style-type: none"> <li>On the toolbar that's above the canvas and to the left, select <b>Horizontal layout</b> . <ul style="list-style-type: none"> <li>Select Vertical layout  to switch back to the original view.</li> </ul> <p>If you're working on a small screen or want to minimize distractions, make the most of your screen's real estate by enlarging the canvas.</p> <li>On the toolbar that's above the canvas and to right, select <b>Maximize</b> . <ul style="list-style-type: none"> <li>Select <b>Minimize</b>  to go back to the original view.</li> </ul> </li> </li></ul> |

If you get stuck while recording, troubleshooting help is available. See [Troubleshoot the Recorder](#).



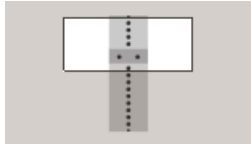
## Tailor Your Building Experience

On the canvas, several settings allow you to tailor your building experience to your preferences. Oracle Integration saves the setting selections in the local storage in your browser.

- Open the robot for which you want to update settings.
  - In the navigation pane, select **Projects**.
  - Select the project name.
  - In the left toolbar, select **Robot** .
  - In the **Robots** box, select the robot to open.

The canvas appears.
- In the title bar of the canvas, next to the Save button, select **Action** , and then select **Settings**.

The Settings panel appears.
- Update the settings as needed to suit your build preferences.

| Tab          | Setting                                       | Description   |
|--------------|---|---|
| Modeling tab | Auto create targets                           | <p>Choose whether to generate a reusable target when you identify a UI control that a robot acts upon.</p> <ul style="list-style-type: none"> <li>When selected and you identify a UI control for a robot action, Oracle Integration creates a reusable target for the UI control, though you can override this setting and hard code the value. If you select the same UI control <i>within the same robot</i>, you can reuse the existing target, create a new target, or hard code the value.</li> <li>When deselected, Oracle Integration hard codes the value for the UI control, though you can override this setting and create a reusable target for the robot.</li> </ul>  |
| Modeling tab | Auto create pre validation page states        | <p>When selected, Oracle Integration creates validation for every robot action based on the field(s) that you select for the action. When the robot runs, this validation ensures that the action starts only after the fields that the robot acts on are visible. This automatic validation is only for before an action occurs. Oracle recommends leaving this setting selected.</p>  |
| Editor tab   | Auto edit newly created actions in the canvas | <p>When selected, after you add an action or logic to a robot, the panel for specifying details opens immediately.</p> <p>Whether to leave this setting selected depends upon your workflow. For instance, if you want to enter all the settings for each action as you build a robot, leave this setting selected. However, if you'd rather add all the actions to the robot and then enter their details later, you'll save some clicks by deselecting this setting.</p>  |
| Editor tab   | Auto close palette drawer after drag and drop | <p>When selected, after you add an action to the canvas using the <b>Robot actions</b> button  on the right toolbar, or after you add logic using the <b>Flow control</b> button , the panel for adding more actions or logic closes.</p> <p>Whether to leave this setting selected depends upon your workflow. For instance, if you want to enter all the settings for each action as you build a robot, deselect this setting. However, if you'd rather add all the actions to the robot and then enter their details later, you'll save some clicks by selecting this setting.</p> <p><b>Note:</b> You typically either select or deselect both of the previous settings. That way, a panel doesn't open over another panel.</p> |
| Editor tab   | Show overview                                 | <p>When selected, the overview box for the canvas appears on canvas. Select the position from the drop-down. The overview box helps you understand which part of the robot you're looking at.</p>   |
| Editor tab   | Show Smart Recording splash screen            | <p>When selected, the splash screen appears when you start using the recorder.</p>  |

4. Select **OK**.

# Create Connections to Applications

A robot must connect to an application to complete its task. Provide the information that the robot needs to connect, such as its credentials and the application's URL, by creating a robot connection. Base each robot connection on a robot connection type, which is similar to a template.

1. [Create a Robot Connection Type](#) (not needed if you can use a predefined connection type)
2. [Create a Robot Connection](#)

## Create a Robot Connection Type


Oracle Integration comes with predefined robot connection types that handle most connection requirements. However, if an application requires additional parameters that aren't included in the predefined types, you can include these parameters in a robot connection type that you create.

To learn more, see [About Robot Connections and Robot Connection Types](#).

### Prerequisites:

- [Create a Project](#)
- Determine whether the predefined robot connection types meet your requirements.  
If so, you don't need to create a new connection type. See [Predefined Robot Connection Types](#).

### To create a robot connection type:

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. In the **Robot connection types** box, click **Add**.  
The Create robot connection type panel appears.
4. Fill in the following fields:
  - **Name:** Enter a name for the robot connection type. When you create a robot connection, you base it on a robot connection type, so enter a helpful and straightforward name. Consider including details about its parameters.
  - **Identifier:** Oracle Integration generates this value using the Name value.
  - **Description:** Provide additional information about the robot connection type.
  - **Keywords:** Enter text that people might use to search for the robot connection type. Press **Enter** after you finish entering each keyword.
5. Add the parameters for the robot connection type.

The parameters are the pieces of data that a robot needs to connection to an application. For example, to connect to a web application, a robot might need an application's URL, credentials, and a key. If you're not sure of the fields the application requires, ask an administrator for the application.

- a. In the panel, next to **Properties**, click **Add +**.
  - b. Fill in the fields:
    - **Name:** Enter a name for the parameter, such as **URL**, **User name**, or **Password**. The value that you provide appears later when you create a robot connection, so enter a helpful and straightforward name.
    - **Type:** Select the data type of the parameter.
    - **Secret:** If the value for the property is a secret value, select this checkbox. For example, passwords are typically secrets. When you select this option, the value is masked; asterisks appear instead of the actual characters.
  - c. Add additional parameters if needed.
6. Click **Create**.

The robot connection type appears in the Robot connection types box with a status of **Configured**. You can now use it to create a robot connection.

Need to update the robot connection type? See [Update a Robot Connection Type](#).


## Create a Robot Connection

A robot connection specifies the information that a robot needs to connect to an application. For example, a typical connection specifies the robot's user name and password, plus the URL for the application.

To learn more, see [About Robot Connections and Robot Connection Types](#).

### Tip:

Oracle recommends reusing robot connections whenever you can. You save time building, and future updates to the values are faster.

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. In the **Robot Connections** box, click **Add**.

The Create robot connection panel appears.
4. In the list, select the robot connection type to base the connection on.

If the robot connection type that you need hasn't been created yet, you can create it. See [Create a Robot Connection Type](#).

The fields that you must define for the robot connection appear, including the parameters that the robot connection type defined.
5. Fill in the following fields:
  - **Name:** Enter a name for the robot connection. Consider including the name of the application for which you're creating the connection.
  - **Identifier:** Oracle Integration generates this value using the Name value.

- **Description:** Provide additional information about the connection.
  - **Keywords:** Enter text that people might use to search for the connection. Press **Enter** after you finish entering each keyword.
6. Enter values for the parameters for the robot connection, such as user name and password.

 **Tip:**

While you build a robot, your robot connection typically uses the URL and credentials for a test, UAT, or similar environment. Later, when you are ready to deploy the robot to a production environment, you can update the robot connection so that it contains the production URL and credentials. See [Update a Robot Connection](#).

7. Click **Create**.

The robot connection appears in the Robot connections box.


If your robot needs to connect to another application, create another connection. Otherwise, start build your robot. See [Create a Robot](#).

## Create a Robot

When you create a robot, you define its trigger and the actions that the robot performs.

To learn more about robots, see [About Robots](#).

**Prerequisite:** [Create a Robot Connection](#).

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. Create a robot.
  - a. In the **Robots** box, click **Add** (if no robots have been built) or **+** (if one or more robots have been built).

The Create robot panel appears.

- b. Enter descriptive metadata for the robot by filling in the following fields:
  - **Name:** Enter a name for the robot. Consider including the action that the robot performs.
  - **Identifier:** Oracle Integration generates this value using the Name value.
  - **Version:** Update the version number, if needed. Most people use the default value.
  - **Description:** Provide additional information about the robot.
  - **Keywords:** Enter text that people might use to search for the robot. Press **Enter** after you finish entering each keyword.
- c. Click **Create**.

The canvas appears. The robot includes an Open Application action that is still undefined.

4. Define the robot's trigger.

A trigger is a JSON object that defines the interface for the robot. You define an input, which is the activity or event that starts the robot. This information comes into the robot. You also define an output, which is the event or incident that the robot produces. This information comes out of the robot after it runs.



a. Select **Click to edit trigger**.

The Trigger panel appears.

b. Define the input: On the **Input** tab, click **Add +**, and fill in the fields.

- **Name:** Enter the name of the input.

For example, if the robot runs after a purchase order is created, you might name the input **PO\_Number**.

When the integration developer calls the robot from an integration, the robot developer maps the input and output to the appropriate fields.

- **Type:** Select the data type of the field.

If the right data type doesn't exist, you can create it. See [Create a Data Type](#).

- **Collection:** Select this value if the input is an array of value. For example, select this option if you're passing an array of PO numbers into the robot.

c. Define another input, if needed.

d. Define the output: Select the **Output** tab, click **Add+**, and fill in the same fields that you completed for the input.

e. Define another output, if needed.

f. Click **OK**.

**Tip:**

If an integration developer needed you to create a robot so that they could start designing the integration that calls the robot, you've now done enough so that they can start this work.

5. Define the open browser action, which is included in every robot.

This action tells the robot to open a web browser and sign in to an application.

For step-by-step instructions, see [Add an Open Browser Action](#).

6. Specify the other steps that the robot completes by adding additional actions and logic to the robot.

See:

- [Quick Start for Building Robots](#)
- [Add an Action to a Robot](#)
- [Add Logic to a Robot](#)

## Add an Action to a Robot

Specify the steps that a robot takes by adding actions to the robot. Each action corresponds to an activity a human would do, such as clicking within a field, entering a value, and submitting the change.

### What Does the Robot Need to Do?

| Type of interaction   | Available actions  |
|---|--|
| Open, close, or switch a browser, or sign in  | <ul style="list-style-type: none"> <li><b>Close browser:</b> Close an internet browser.</li> <li><b>Login:</b> Sign in to an application.</li> <li><b>Open browser:</b> Open an internet browser and, optionally, sign in to an application.</li> <li><b>Switch browser:</b> Start working in a different internet browser tab or window.</li> </ul>       |
| Select an element in a user interface   | <ul style="list-style-type: none"> <li><b>Checkbox:</b> Select or deselect a checkbox, or determine whether a checkbox is selected.</li> <li><b>Click element:</b> Select a button, link, some items in lists, and some checkboxes.</li> <li><b>List:</b> Select one or more items in a list or drop-down list.</li> </ul>                                 |
| Interact with text in a user interface, such as obtaining, entering, or clearing text | <ul style="list-style-type: none"> <li><b>Clear text:</b> Clear text from a field.</li> <li><b>Define web table:</b> Get text from one or more columns in a table.</li> <li><b>Enter text:</b> Enter text into a field.</li> <li><b>Get text:</b> Get text from the user interface.</li> <li><b>Radio button:</b> Interact with a radio button.</li> </ul> |
| Add testing and validation actions  | <ul style="list-style-type: none"> <li><b>Log:</b> Record an entry in the activity stream.</li> <li><b>Screenshot:</b> Capture a screenshot and save it as a file.</li> <li><b>Wait until element is visible:</b> Pause until an element in the user interface is visible.</li> </ul>  |
| Manage variables  | <b>Data stitch:</b> Manage the variables in a robot.   |

### Get Help with the Settings for Each Robot Action

See [Deeper Dive: Settings for Robot Actions](#).

## Add a Checkbox Action

The checkbox action interacts with a checkbox, such as by selecting it, deselecting it, or determining whether it's selected.

You can use the checkbox action only for a UI element with a tag name of `INPUT` and a type attribute of `checkbox`.




#### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

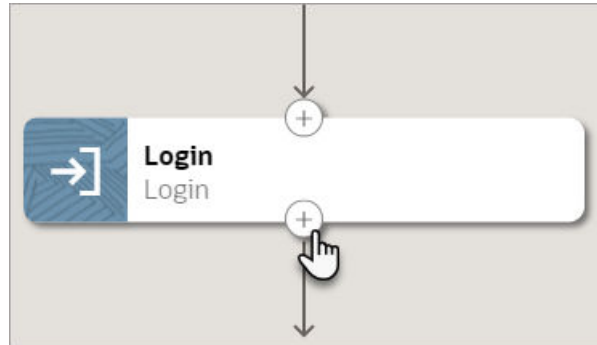
1. Open the robot to edit.




- a. In the navigation pane, select **Projects**.
- b. Select the project name.
- c. In the left toolbar, select **Robot** .
- d. In the **Robots** box, select the robot to open.  
The canvas appears.

2. Add the action to the robot.

- a. On the canvas, point to an action, and click **+**.

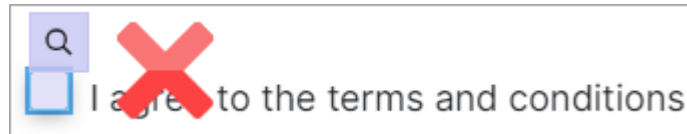


A menu of available actions appears.

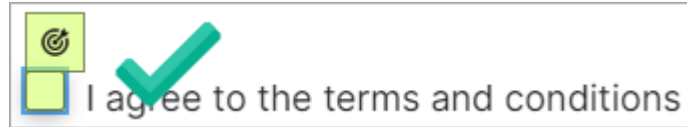
- b. Select **Checkbox**.  
A Checkbox action appears on the canvas, and the Checkbox panel appears.
3. In the panel, enter a **Name** and **Description** for the action.  
The Name appears on the action in the canvas and should help you and others understand the goal of the action.
4. From **Operations**, select how the robot interacts with the checkbox.
  - **Select Checkbox:** The robot selects a checkbox.
  - **Unselect Checkbox:** The robot clears a checkbox selection.
  - **Is Checkbox Selected:** The robot checks whether a checkbox is selected. This option is useful if you need to take a different action depending on whether a checkbox is selected.
5. On the **Input** tab, specify input details for the action.
  - a. In another browser window, open the application that the robot needs to work in.
  - b. In the robot, click within the **Locator** field, and select **Target a page element** .

The Target a page element panel appears.

  - c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.  
The application that the robot needs to work in opens.
  - d. In the application that the robot needs to work in, point to the checkbox that the robot needs to interact with, but don't select the checkbox yet.  
For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.




For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.


6. On the **Input** tab, specify whether to capture any screenshots as part of the action.  
See [Capture Screenshots in Robots](#).
7. If you selected the **Is Checkbox Selected** operation, specify where to save the answer that you get on the **Output** tab.
  - Assign the value to a variable.
    - a. Click within the **Save to** field, and select **Variables** (x).  
The Variables panel appears.
    - b. Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#).
    - c. Select the variable to assign the value to, and drag it to the **Save to** field.
  - Assign the value to a property of the output property.
    - a. Click within the **Save to** field, select **More options**, and then select  **Output**.  
The Output panel appears.
    - b. Determine whether the output property that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
    - c. Select the output property to assign the value to, and drag it to the **Save to** field.
8. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.  
See [Add Validation to a Robot Action](#).
9. Click **OK**.
10. Above the canvas, select **Save**.

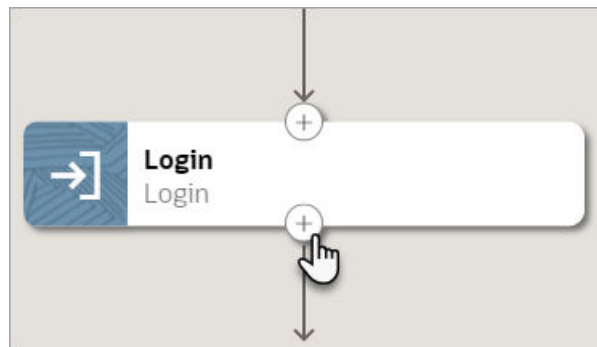
## Add a Clear Text Action

The clear text action removes text from a field in a user interface.


### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.

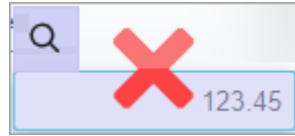


A menu of available actions appears.

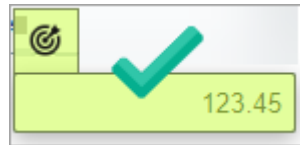
- b. Select **Clear Text**.  
A Clear Text action appears on the canvas, and the Clear Text panel appears.
3. In the panel, enter a **Name** and **Description** for the action.  
The Name appears on the action in the canvas and should help you and others understand the goal of the action.
  4. On the **Input** tab, specify input details for the action.
    - a. In another browser window, open the application that the robot needs to work in.
    - b. In the robot, click within the **Locator** field, and select **Target a page element** .
    - The Target a page element panel appears.
    - c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.  
The application that the robot needs to work in opens.

- d. In the application that the robot needs to work in, point to the field from which the robot needs to clear text, but don't select the field yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

5. On the **Input** tab, specify whether to capture any screenshots as part of the action.  
See [Capture Screenshots in Robots](#).
6. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.  
See [Add Validation to a Robot Action](#).
7. Click **OK**.
8. Above the canvas, select **Save**.

## Add a Click Element

The click element action selects any element in a user interface, such as a button or link.



Several other clicking-related action are available and can offer more control, including the [checkbox](#) and [list](#) actions.

Consider creating any robot resources that the robot action will use, such as variables or a trigger's input and output. If you choose not to create them in advance, you can pause the recorder and create them as you build.

## Add a Click Element Using the Recorder

The recorder offers an intuitive experience for easy building.

1. Open the robot to edit.

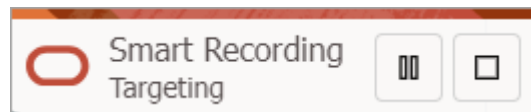
- a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Start the recorder.
    - a. On the canvas, select the action that you want to record after.
    - b. On the toolbar, select **Record after the selected action** .
    - c. In the Smart record panel, open the **Select browser tab to target** drop-down, and select the application that you want to work in. You might need to scroll down to find the application.

If the application doesn't appear in the list, close the Smart record panel, open the application in another tab, and start the recorder again.

A splash screen appears, and then the RPA Smart Recording panel appears.

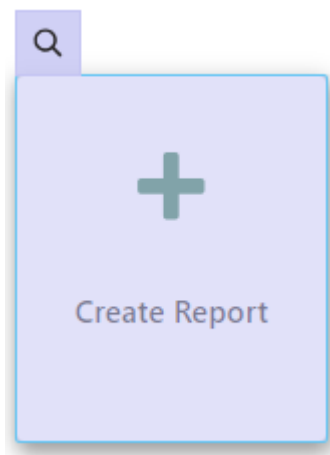
- d. Select **Begin Recording**.

The Smart Recording window appears in the lower-left corner of your browser. Additionally, your mouse cursor can now target elements in the user interface.

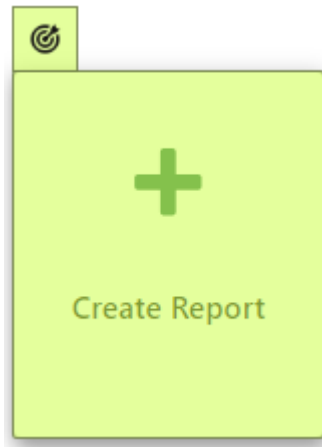


3. Identify the element in the user interface that the robot needs to interact with.
  - a. Point to the field from which the robot needs to click a UI element, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- b. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The Smart Recorder panel appears with details about the element you selected.

c. Review and update the fields as needed.

- **Name:** Enter the name of the target. This text appears on the robot action in the canvas and in the list of targets that are in the robot.

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.


- **Target name:** Review the XML Path Language, or XPath, for the element that you selected. Underscores ( `_` ) appear in place of invalid characters. You can update the value, if you want.
- **Element:** Review the HTML element that you selected. If you selected the wrong element type, select **Discard** in the panel, and select a different element.
- **Action:** Select **Click Element**.
- **Execute action on save:** Select this option if you want to complete the action that you just recorded. For example, if you just clicked a button and you want the click to occur in the application after you save these changes, select this option.


d. Select **Save**.

4. Choose the appropriate next step:


- Add another action using the recorder.

See [Add an Action to a Robot](#).

- To pause the recorder so you can figure out your next steps, select **Pause**  in the Smart Recording window in the lower-left corner of the browser.

- To stop the recorder and return to the canvas, select **Stop**  in the Smart Recording window in the lower-left corner of the browser.

If you close the application that you're recording in, you can still stop the recorder.

Select **Stop**  in the toolbar of the canvas.


- To customize the action you just added, such as by creating validation or identifying the screenshots to capture, stop the recording, double-click the action on the canvas, and update the action as needed.

All actions are read-only until you stop the recorder.

- Above the canvas, select **Save**.

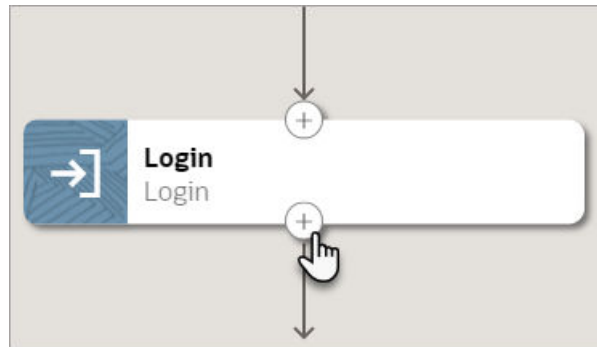
## Add a Click Element Using the Low-Code Tools

The low-code tools offer enhanced control.


- Open the robot to edit.
  - In the navigation pane, select **Projects**.
  - Select the project name.
  - In the left toolbar, select **Robot** .
  - In the **Robots** box, select the robot to open.

The canvas appears.

- Add the action to the robot.
  - On the canvas, point to an action, and click **+**.



A menu of available actions appears.

- Select **Click Element**.  
A Click Element action appears on the canvas, and the Click Element panel appears.
- In the panel, enter a **Name** and **Description** for the action.  
The Name appears on the action in the canvas and should help you and others understand the goal of the action.
  - On the **Input** tab, specify input details for the action.
    - In another browser window, open the application that the robot needs to work in.
    - In the robot, click within the **Locator** field, and select **Target a page element** .

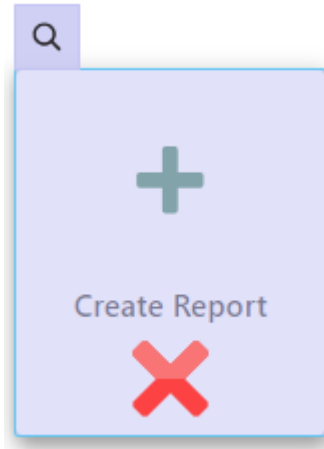
The Target a page element panel appears.

- In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

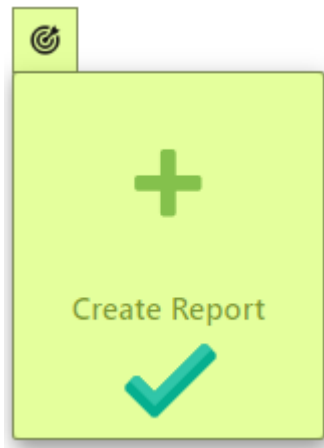
The application that the robot needs to work in opens.

- d. In the application that the robot needs to work in, point to the field from which the robot needs to click a UI element, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

5. On the **Input** tab, specify whether to capture any screenshots as part of the action. See [Capture Screenshots in Robots](#).
6. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action. See [Add Validation to a Robot Action](#).



7. Click **OK**.
8. Above the canvas, select **Save**.


## Add a Close Browser Action

The close browser action closes the web browser that the robot is working in. You typically use the close browser action with the switch browser action.

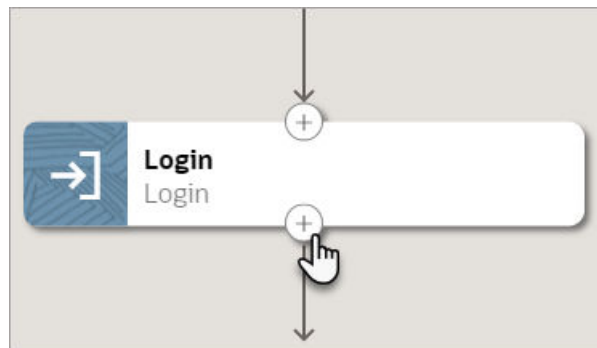
For example, you might get data from one browser, and then switch to a different browser window to paste the values into the application or get more data.

### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

- b. Select **Close Browser**.

A Close Browser action appears on the canvas, and the Close Browser panel appears.
3. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.
4. Click **OK**.
5. Above the canvas, select **Save**.

## Add a Data Stitch Action

A data stitch lets you perform an action on the data that a robot collects. With a data stitch, you can move data across variables and output properties.

Unlike other robot actions, a data stitch doesn't allow a robot to interact with an application's user interface. Instead, it allows you to manipulate data. For example, you can build an array of data or a list structure by assigning and appending values for variables and output properties.

### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

## Use Case: Assigning a Value

The assign operation for a data stitch action assigns a new value to a variable.

For example, consider a robot that needs to provide the total value of an invoice. The robot uses the get text action to obtain two values: an invoice value and a tax value. The robot assigns the values to two variables.

Next, the robot must add these values to calculate the total value of the invoice. To add these values, create an expression and assign the total to an output variable. You perform both of these tasks in the data stitch action.

## Use Case: Appending a Value


The append operation for a data stitch action preserves the existing value for a variable and appends a new value as a suffix. Use the append operation to build a collection as a response or output from a robot.

For example, consider a robot that updates a set of invoices. Use a foreach loop to update each invoice, one at a time. To provide the robot with the invoice numbers to update, define the input's trigger, which is a collection that holds arrays of values for each property.

After the foreach loop processes each invoice, the data stitch action inserts the invoice number into another variable.

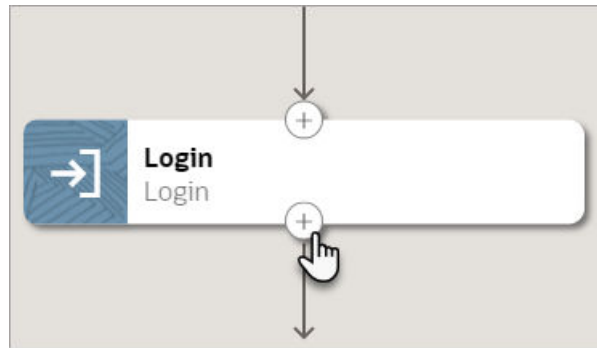
For more details about this use case, see [Use Case: Save Data After Iterating](#).

## Add a Data Stitch Action Using the Low-Code Tools

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.

2. Add the action to the robot.
  - a. On the canvas, point to an action, and click +.



A menu of available actions appears.

- b. Select **Data Stitch**.

A data stitch action appears on the canvas, and the Data Stitch panel appears.
  3. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.
  4. Specify how you need to manipulate data.
    - a. Select **Add assignment**.

The cursor appears in the Variable field, and the Variables panel appears.
    - b. Drag a variable to the **Variable** field.

The data stitch action assigns or appends a value to a variable. This variable receives the value.
    - c. From the **Operation** drop-down, select one of the following options:
      - **Assign**: Select this option when the robot assigns a new value to the variable that you selected.
      - **Append**: Select this option when the robot preserves the existing value for a variable and appends a new value as a suffix.
    - d. In the **Value** field, enter the value to assign or append to the variable that you selected for the **Variable** field.

For example, you might drag a variable to the field, hard code a value, or compute a value using an [expression](#).
5. Add additional assignments to the data stitch as needed.

To see a sample use case, see [Use Case: Save Data After Iterating](#).
6. Click **OK**.
7. Above the canvas, select **Save**.

## Add a Define Web Table Action

The define web table action allows you to identify the columns in a table that you are interested in. After you add a define web table action, you can add more actions, depending on the work that the robot needs to complete for the table.

### Use Case

Consider a table that contains invoice numbers in one column and a hyperlink in another column. You can add a get text action to get the invoice numbers and a click element action to click the link. Use a [foreach loop](#) to iterate on multiple rows of data.

You can use the read data table action only for a UI element that is coded as a `table`.

### Output of the Action

The define web table returns an output: A collection of HTML table rows for the columns that you select. Each row consists of key-value pairs:


- The key is the column name.
- The value is the xpath of the corresponding row cell.  
You can use the xpath later, such as in a get text or click action.

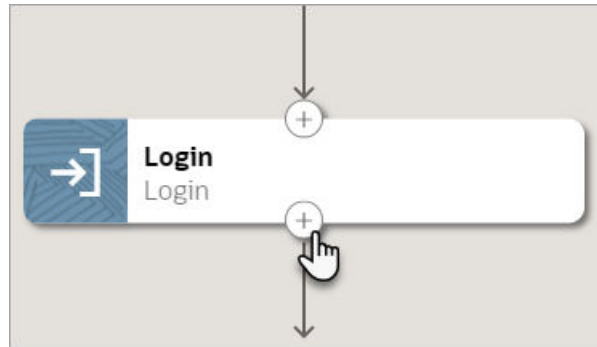
### Add a Define Web Table Action:



#### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Verify that the table in the application is coded as a `table` element in the page's HTML.  
For help understanding the HTML of a page, see [View the HTML Code for a Page](#).  
If the table was coded using `div` elements and the page's CSS transforms the content into a table, you can't use this action. Instead, use the get text action and save the individual values as a variable collection. See [Add a Get Text Action](#).
2. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
3. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

b. Select **Define Web Table**.

A Define Web Table action appears on the canvas, and the Define Web Table panel appears.


4. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.

5. On the **Input** tab, identify the first column of data that the robot must interact with.

Remember that the first column that the robot interacts with might or might not be the first column in the table. In the define web table action, you don't need to identify every column in the table. Instead, just identify the columns that the robot interacts with.

a. In another browser window, open the application that the robot needs to work in.

b. Click within the **Header** field, and select **Target a page element** .

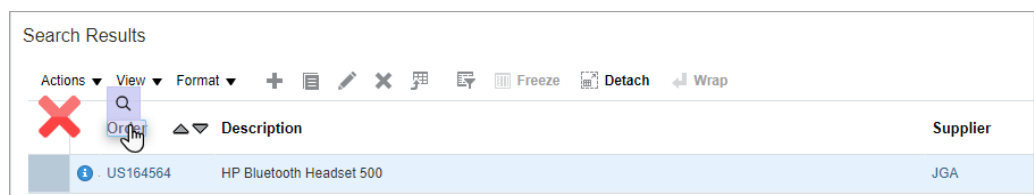
The Target a page element panel appears.

c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

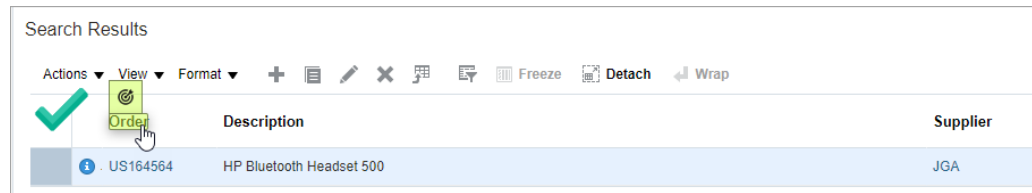
The application that the robot needs to work in opens.

d. In the application that the robot needs to work in, point to the header of the first column that the robot needs to interact with, but don't select the column header yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.




For more tips, see [Quick Start for Building Robots](#).



The recorder enters a value in the **Header** field in the robot.

### Tip:

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

- f. Click within the **Row 1** field, and repeat the previous steps to target the first row in the column to target.
  - g. Click within the **Row 2** field, and repeat the previous steps to target the second row in the column to target.
6. If the robot must interact with one or more additional columns in the table, identify the additional columns on the **Input** tab.
- a. Select **Add** .
  - b. Repeat the previous steps to identify the header and first row of the column that the robot interacts with.
  - c. Add more columns as needed until you've identified all the columns that the robot interacts with.
7. On the **Output** tab, review the variable name that the table information is saved to.
- The define web table creates a custom data type and variable. The properties of the data type are the columns that you identified in the define web table action. The variable is a collection of the data type that the define web table action created.
8. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.
- See [Add Validation to a Robot Action](#).
9. Click **OK**.
10. Above the canvas, select **Save**.

Your next steps depend upon your goals:

- To iterate over the records in the table, [add a foreach loop](#).
- To record the values from the table to the activity stream, [add a log action](#).
- To view the data type or variable that the define web table action created, select **Data types**  or **Variables**  on the toolbar in the canvas.



## Add an Enter Text Action

The enter text action inserts a value into a field in a user interface.

You typically use the enter text action in a text field that is coded as an `<input>` field in the page's HTML. The robot can enter a variable, an input parameter, or a hard-coded value into the field.

## Add an Enter Text Action Using the Recorder

Consider creating any robot resources that the robot action will use, such as variables or a trigger's input and output. If you choose not to create them in advance, you can pause the recorder and create them as you build.

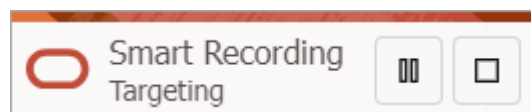
1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Start the recorder.
  - a. On the canvas, select the action that you want to record after.
  - b. On the toolbar, select **Record after the selected action** .
  - c. In the Smart record panel, open the **Select browser tab to target** drop-down, and select the application that you want to work in. You might need to scroll down to find the application.

If the application doesn't appear in the list, close the Smart record panel, open the application in another tab, and start the recorder again.

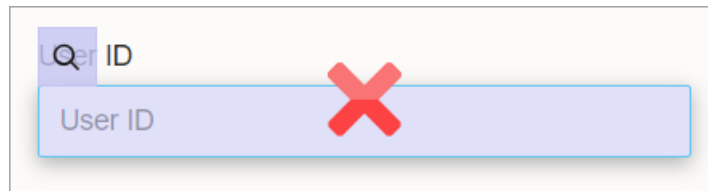
A splash screen appears, and then the RPA Smart Recording panel appears.

- d. Select **Begin Recording**.

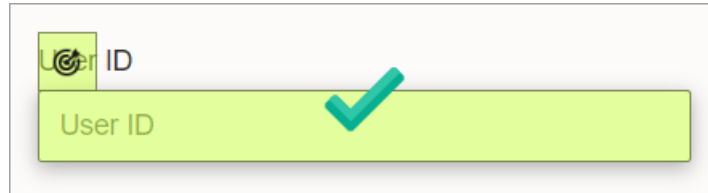
The Smart Recording window appears in the lower-left corner of your browser. Additionally, your mouse cursor can now target elements in the user interface.



3. Identify the element in the user interface that the robot needs to interact with.
  - a. Point to the field from which the robot needs to click a UI element, but don't select the element yet.  
For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- b. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).


The Smart Recorder panel appears with details about the element you selected.


- c. Review and update the fields as needed.
- **Name:** Enter the name of the target. This text appears on the robot action in the canvas and in the list of targets that are in the robot.  
Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.
  - **Target name:** Review the XML Path Language, or XPath, for the element that you selected. Underscores ( `_` ) appear in place of invalid characters. You can update the value, if you want.
  - **Element:** Review the HTML element that you selected. If you selected the wrong element type, select **Discard** in the panel, and select a different element.
  - **Action:** Select **Enter Text**.
  - **Value:** Enter the value that the robot should insert into the field.

The following instructions are for selecting a variable. To review all of your options for the value, see [Define the Fields of an Action](#).


- Select within the **Value** field, and select **Variables** (x) .  
The Variables panel appears.
  - If the variable to use doesn't exist yet, create it. See [Create a Variable](#).
  - From the Available variables list, select a variable, and drag it to the **Value** field.
- **Test value:** Provide the value that the recorder should enter into the field so that you can continue with the recording.
- d. Select **Save**.
4. Choose the appropriate next step:
- Add another action using the recorder.  
See [Add an Action to a Robot](#).



- To pause the recorder so you can figure out your next steps, select **Pause**  in the Smart Recording window in the lower-left corner of the browser.

- To stop the recorder and return to the canvas, select **Stop**  in the Smart Recording window in the lower-left corner of the browser.

If you close the application that you're recording in, you can still stop the recorder.


Select **Stop**  in the toolbar of the canvas.

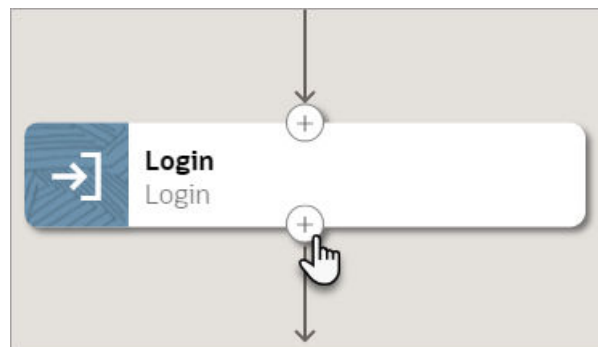
- To customize the action you just added, such as by creating validation or identifying the screenshots to capture, stop the recording, double-click the action on the canvas, and update the action as needed.

All actions are read-only until you stop the recorder.

- Above the canvas, select **Save**.

## Add an Enter Text Action Using the Low-Code Tools

- Open the robot to edit.
  - In the navigation pane, select **Projects**.
  - Select the project name.
  - In the left toolbar, select **Robot** .
  - In the **Robots** box, select the robot to open.  
The canvas appears.
- Add the action to the robot.
  - On the canvas, point to an action, and click **+**.



A menu of available actions appears.


- Select **Enter Text**.

An Enter Text action appears on the canvas, and the Enter Text panel appears.

- In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.

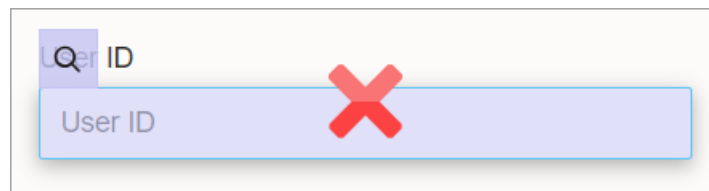
- On the **Input** tab, specify input details for the action.

- a. In another browser window, open the application that the robot needs to work in.
- b. In the robot, click within the **Locator** field, and select **Target a page element** . The Target a page element panel appears.
- c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

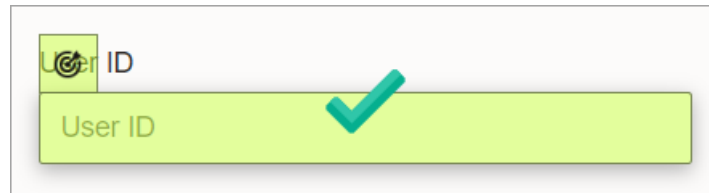
The application that the robot needs to work in opens.

- d. In the application that the robot needs to work in, point to the field into which the robot needs to enter text, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.




For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

- f. In the **Text** field, enter the value that the robot should insert into the field. The following instructions are for selecting a variable. To review all of your options for the value, see [Define the Fields of an Action](#).
    - i. Select within the **Text** field, and select **Variables** . The Variables panel appears.
    - ii. If the variable to use doesn't exist yet, create it. See [Create a Variable](#).
    - iii. From the Available variables list, select a variable, and drag it to the **Text** field.
5. On the **Input** tab, specify whether to capture any screenshots as part of the action.

See [Capture Screenshots in Robots](#).

6. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.

See [Add Validation to a Robot Action](#).

7. Click **OK**.
8. Above the canvas, select **Save**.

## Add a Get Text Action

The get text action collects a value from a user interface and then assigns the value to a variable or an output parameter.

### Use Cases



Consider a scenario in which a robot must verify that the invoice totals in a report are identical to the actual totals of the invoices.

After the robot uses the get text action to obtain the total of each invoice, you can take the following actions:

- **Report the actual total of each invoice:** To complete this task, assign the value for an invoice total to an output parameter.
- **Complete additional processing on the invoice totals:** To complete this task, assign the value for an invoice total to a variable.

## Add a Get Text Action Using the Recorder

Consider creating any robot resources that the robot action will use, such as variables or a trigger's input and output. If you choose not to create them in advance, you can pause the recorder and create them as you build.

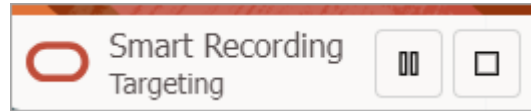
1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Start the recorder.
  - a. On the canvas, select the action that you want to record after.
  - b. On the toolbar, select **Record after the selected action** .
  - c. In the Smart record panel, open the **Select browser tab to target** drop-down, and select the application that you want to work in. You might need to scroll down to find the application.

If the application doesn't appear in the list, close the Smart record panel, open the application in another tab, and start the recorder again.

A splash screen appears, and then the RPA Smart Recording panel appears.

d. Select **Begin Recording**.

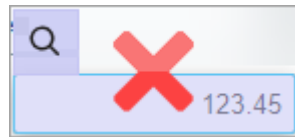
The Smart Recording window appears in the lower-left corner of your browser. Additionally, your mouse cursor can now target elements in the user interface.



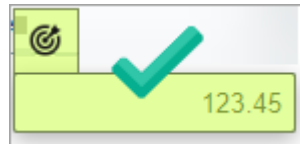
## 3. Identify the element in the user interface that the robot needs to interact with.

- a. Point to the field from which the robot needs to click a UI element, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- b. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.

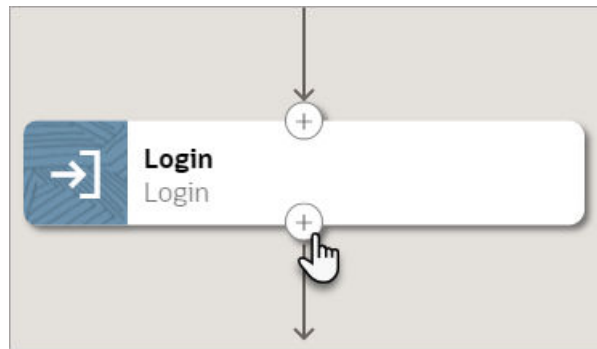


For more tips, see [Quick Start for Building Robots](#).

The Smart Recorder panel appears with details about the element you selected.

- c. Review and update the fields as needed.
- **Name:** Enter the name of the target. This text appears on the robot action in the canvas and in the list of targets that are in the robot.  
Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.
  - **Target name:** Review the XML Path Language, or XPath, for the element that you selected. Underscores ( `_` ) appear in place of invalid characters. You can update the value, if you want.
  - **Element:** Review the HTML element that you selected. If you selected the wrong element type, select **Discard** in the panel, and select a different element.
  - **Action:** Select **Get Text**.
- d. For **Save to**, specify where to save the text that you get from the field. You have the following options:
- Assign the value to a variable.
    - i. Click within the **Save to** field, and select **Variables** (x).  
The Variables panel appears.





A menu of available actions appears.

- b. Select **Get Text**.


A Get Text action appears on the canvas, and the Get Text panel appears.

3. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.

4. On the **Input** tab, specify input details for the action.

- a. In another browser window, open the application that the robot needs to work in.

- b. In the robot, click within the **Locator** field, and select **Target a page element** .

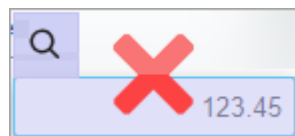
The Target a page element panel appears.

- c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

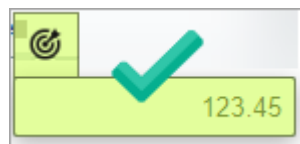
The application that the robot needs to work in opens.

- d. In the application that the robot needs to work in, point to the field from which the robot needs to get text, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.




For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

5. On the **Input** tab, specify whether to capture any screenshots as part of the action. See [Capture Screenshots in Robots](#).
6. On the **Output** tab, specify where to save the text that you get from the field. You have the following options:
  - Assign the value to a variable.
    - a. Click within the **Save to** field, and select **Variables (x)**.  
The Variables panel appears.
    - b. Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#).
    - c. Select the variable to assign the value to, and drag it to the **Save to** field.
  - Assign the value to a property of the output property.
    - a. Click within the **Save to** field, select **More options**, and then select  **Output**.  
The Output panel appears.
    - b. Determine whether the output property that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
    - c. Select the output property to assign the value to, and drag it to the **Save to** field.
7. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action. See [Add Validation to a Robot Action](#).
8. Click **OK**.
9. Above the canvas, select **Save**.

## Add a List Action

The list action interacts with a list, such as by selecting, deselecting, or getting the value of one or more items.



## Use Cases

This action has specific requirements for the page's underlying HTML code.

You can use this action only when a UI element has a web element of `select`. For drop-down and multi-select lists, the HTML code might wrap the `select` element in an `option` element.

## Add a List Action Using the Recorder

1. Open the robot to edit.

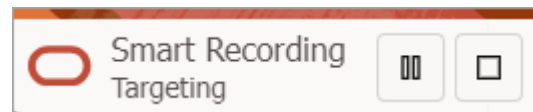
- a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Start the recorder.
- a. On the canvas, select the action that you want to record after.
  - b. On the toolbar, select **Record after the selected action** .
  - c. In the Smart record panel, open the **Select browser tab to target** drop-down, and select the application that you want to work in. You might need to scroll down to find the application.

If the application doesn't appear in the list, close the Smart record panel, open the application in another tab, and start the recorder again.

A splash screen appears, and then the RPA Smart Recording panel appears.

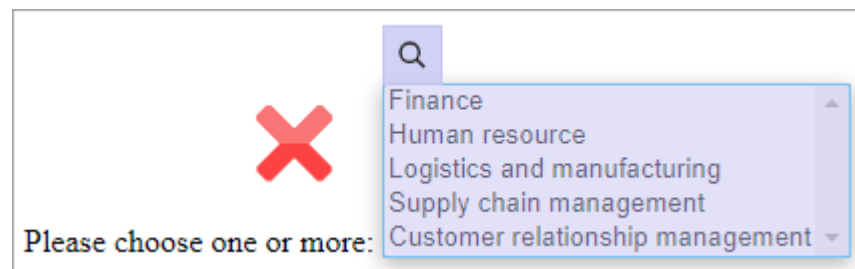
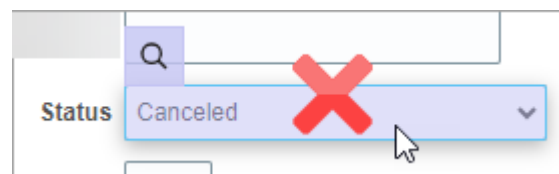
- d. Select **Begin Recording**.

The Smart Recording window appears in the lower-left corner of your browser. Additionally, your mouse cursor can now target elements in the user interface.



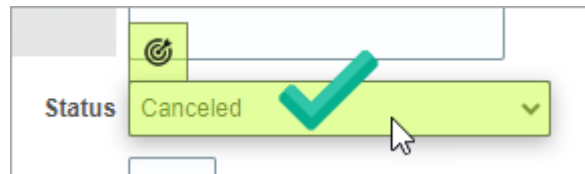
3. Identify the element in the user interface that the robot needs to interact with.
- a. Point to the list that the robot needs to interact with, but don't select the list yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- b. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.





For more tips, see [Quick Start for Building Robots](#).

The Smart Recorder panel appears with details about the element you selected.

c. Review and update the fields as needed.



- **Name:** Enter the name of the target. This text appears on the robot action in the canvas and in the list of targets that are in the robot.




Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

- **Target name:** Review the XML Path Language, or XPath, for the element that you selected. Underscores ( `_` ) appear in place of invalid characters. You can update the value, if you want.
- **Action:** Select how the robot interacts with the list.


| Option                   | Description  | Selection list type     |
|--------------------------|--|-------------------------|
| Get Selected List Label  | Get the HTML label attribute of the selected item in a list.<br>If no items are selected, the robot returns an empty value.          | Single Selection List   |
| Get Selected List Labels | Get the HTML label attribute of each item that is selected in a list.<br>If no items are selected, the robot returns an empty value. | Multiple Selection List |
| Get Selected List Value  | Get the HTML value attribute of the selected item in a list.<br>If no items are selected, the robot returns an empty value.          | Single Selection List   |
| Get Selected List Values | Get the HTML value attribute of each item that is selected in a list.<br>If no items are selected, the robot returns an empty value. | Multiple Selection List |

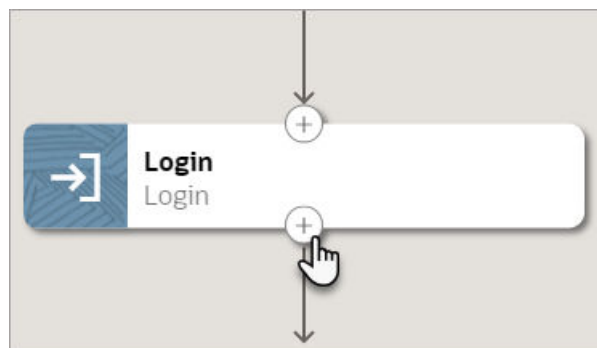
| Option                      | Description   | Selection list type                              |
|-----------------------------|---|--|
| Select From List By Index   | Select one or more items in a list according to the HTML index attribute that you provide.<br><br>The indexes of list options start from 0. That means that the first entry in the list is numbered 0, the second entry is numbered 1, and so on.   | Single Selection List<br>Multiple Selection List |
| Select From List By Label   | Select one or more items in a list according to the HTML label attribute that you provide.  | Single Selection List<br>Multiple Selection List |
| Select From List By Value   | Select one or more items in a list according to the HTML value attribute that you provide.  | Single Selection List<br>Multiple Selection List |
| Unselect From List By Index | Deselect one or more items in a list according to the HTML index attribute that you provide.<br><br>The indexes of list options start from 0. That means that the first entry in the list is numbered 0, the second entry is numbered 1, and so on. | Multiple Selection List                          |
| Unselect From List By Label | Deselect one or more items in a list according to the HTML label attribute that you provide.  | Multiple Selection List                          |
| Unselect From List By Value | Deselect one or more items in a list according to the HTML value attribute that you provide.  | Multiple Selection List                          |

- d. For **Save to**, specify where to save the text that you get from the field. You have the following options:
- Assign the value to a variable.
    - i. Click within the **Save to** field, and select **Variables** . The Variables panel appears.
    - ii. Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#).
    - iii. Select the variable to assign the value to, and drag it to the **Save to** field.
  - Assign the value to a property of the output property.
    - i. Click within the **Save to** field, select **More options**, and then select  **Flow Input/Output**. The Input & Output panel appears.
    - ii. Select the **Output** tab.
    - iii. Determine whether the output property that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).

- iv. Select the output property to assign the value to, and drag it to the **Save to** field.
  - e. Select **Save**.
4. Choose the appropriate next step:
  - Add another action using the recorder.  
See [Add an Action to a Robot](#).
  - To pause the recorder so you can figure out your next steps, select **Pause**  in the Smart Recording window in the lower-left corner of the browser.
  - To stop the recorder and return to the canvas, select **Stop**  in the Smart Recording window in the lower-left corner of the browser.  
If you close the application that you're recording in, you can still stop the recorder.  
Select **Stop**  in the toolbar of the canvas.
  - To customize the action you just added, such as by creating validation or identifying the screenshots to capture, stop the recording, double-click the action on the canvas, and update the action as needed.  
All actions are read-only until you stop the recorder.
5. Above the canvas, select **Save**.

## Add a List Action Using the Low-Code Tools

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

**b. Select List.**

A List action appears on the canvas, and the List panel appears.

**3. In the panel, enter a Name and Description for the action.**

The Name appears on the action in the canvas and should help you and others understand the goal of the action.


**4. From the Selection List Type drop-down, select the type of list that the robot interacts with.**

| Option                  | Description   |
|-------------------------|---|
| Single Selection List   | The list allows you to select only one option.<br>If the robot will select only one option, but the list allows multiple items to be selected, select <b>Multiple Selection List</b> instead. |
| Multiple Selection List | The list allows you to select one or more options.  |

**5. From Operations, select how the robot interacts with the list.**

| Option                    | Description   | Selection list type                              |
|---------------------------|---|--|
| Get Selected List Label   | Get the HTML label attribute of the selected item in a list.<br>If no items are selected, the robot returns an empty value.   | Single Selection List                            |
| Get Selected List Labels  | Get the HTML label attribute of each item that is selected in a list.<br>If no items are selected, the robot returns an empty value.  | Multiple Selection List                          |
| Get Selected List Value   | Get the HTML value attribute of the selected item in a list.<br>If no items are selected, the robot returns an empty value.   | Single Selection List                            |
| Get Selected List Values  | Get the HTML value attribute of each item that is selected in a list.<br>If no items are selected, the robot returns an empty value.  | Multiple Selection List                          |
| Select From List By Index | Select one or more items in a list according to the HTML index attribute that you provide.<br>The indexes of list options start from 0. That means that the first entry in the list is numbered 0, the second entry is numbered 1, and so on. | Single Selection List<br>Multiple Selection List |
| Select From List By Label | Select one or more items in a list according to the HTML label attribute that you provide.  | Single Selection List<br>Multiple Selection List |
| Select From List By Value | Select one or more items in a list according to the HTML value attribute that you provide.  | Single Selection List<br>Multiple Selection List |

| Option                      | Description   | Selection list type     |
|-----------------------------|---|-------------------------|
| Unselect From List By Index | Deselect one or more items in a list according to the HTML index attribute that you provide.<br><br>The indexes of list options start from 0. That means that the first entry in the list is numbered 0, the second entry is numbered 1, and so on. | Multiple Selection List |
| Unselect From List By Label | Deselect one or more items in a list according to the HTML label attribute that you provide.  | Multiple Selection List |
| Unselect From List By Value | Deselect one or more items in a list according to the HTML value attribute that you provide.  | Multiple Selection List |

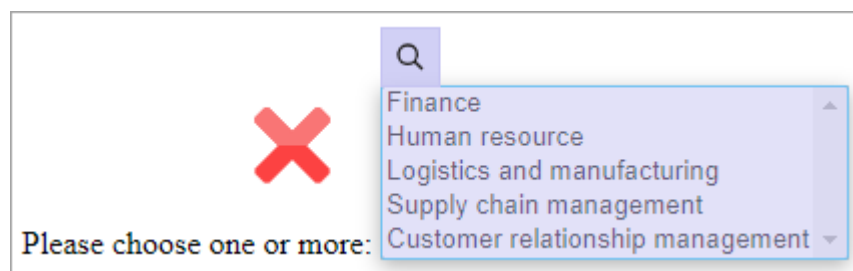
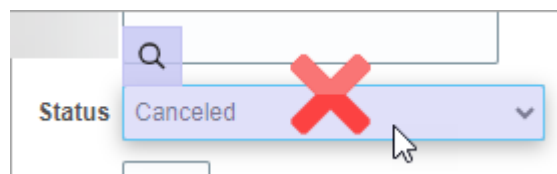
6. On the **Input** tab, specify input details for the action.
  - a. In another browser window, open the application that the robot needs to work in.
  - b. In the robot, click within the **Locator** field, and select **Target a page element** .

The Target a page element panel appears.
  - c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

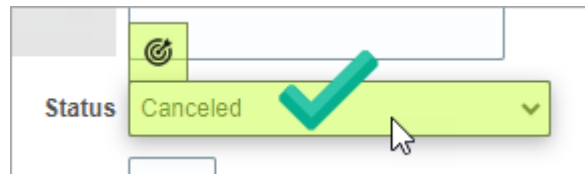
The application that the robot needs to work in opens.
  - d. In the application that the robot needs to work in, point to the list that the robot needs to interact with, but don't select the list yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.

Regardless of the operation that you chose, always point to the list itself.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**


Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

7. On the **Input** tab, specify whether to capture any screenshots as part of the action. See [Capture Screenshots in Robots](#).

If you choose an operation that gets a label or value, Oracle recommends taking a screenshot of the list.

8. On the **Output** tab, specify where to save the value(s) that you got from the action.

The Output tab appears for only some operations, including **Get Selected List Label(s)** and **Get Selected List Value(s)**.

- Assign the value to a variable.
  - a. Click within the **Save to** field, and select **Variables (x)**.  
The Variables panel appears.
  - b. Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#).
  - c. Select the variable to assign the value to, and drag it to the **Save to** field.  
If the action interacts with a multiselect list, you must select a collection variable, even if the operation gets a single value. If the action interacts with a single-select list, you must select a non-collection variable. Otherwise, an error occurs for the action when you save the robot.
- Assign the value to a property of the output property.
  - a. Click within the **Save to** field, select **More options**, and then select  **Output**.  
The Output panel appears.

- b. Determine whether the output property that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
  - c. Select the output property to assign the value to, and drag it to the **Save to** field.  
If the action interacts with a multiselect list, you must select a collection variable, even if the operation gets a single value. If the action interacts with a single-select list, you must select a non-collection variable. Otherwise, an error occurs for the action when you save the robot.
9. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.  
See [Add Validation to a Robot Action](#).
  10. Click **OK**.
  11. Above the canvas, select **Save**.


## Add a Log Action

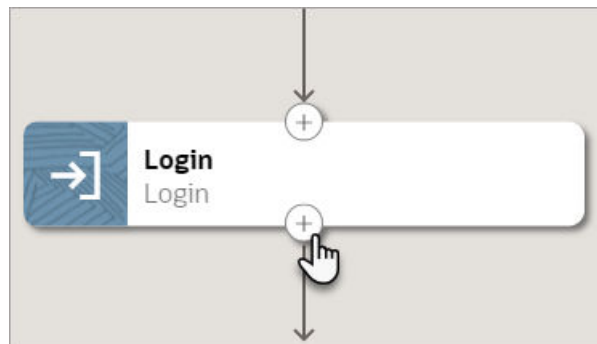
The log action adds a message to the activity stream for a robot instance. Unlike other robot actions, the log action doesn't interact with the user interface.

You can record any message to the activity stream, but you typically record a value that the robot obtains. For instance, you might record the value from a get text action. The value could be the output of the robot or a value that the robot obtains and later updates using logic.

### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

- b. Select **Log**.

A Log action appears on the canvas, and the Log panel appears.

3. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.

4. In the **Message** field, enter the message to record in the activity stream.

The message can include some or all of the following information:

- Typed text
- Variables
- User interface element
- Input and output properties from the trigger
- Fields from a robot connection

For details on how to include these values in a message, see [Define the Fields of an Action](#).


5. Click **OK**.
6. Above the canvas, select **Save**.

## Add a Login Action

The login action enters a robot's user name and password and signs the robot in to an application.

### **Note:**

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.


The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.

A menu of available actions appears.





The Robot connections panel appears.

- c. In the list, expand the entry for a robot connection.
  - d. Drag a parameter for the robot connection to the field.
  - e. Repeat these steps for the **Password** field, which holds the robot's password for the application.
5. On the **Input** tab, specify input details for the action.
- a. In another browser, open the sign-in page for the application that the robot will work in.
  - b. In the robot, select within the **Username locator** field, and select **Target a page element** .

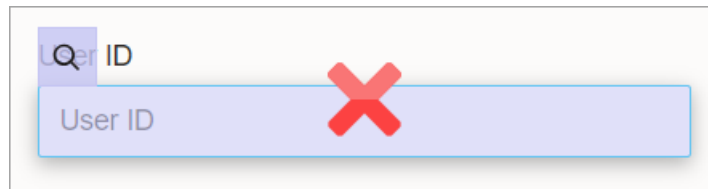
The Target a page element panel appears.

- c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

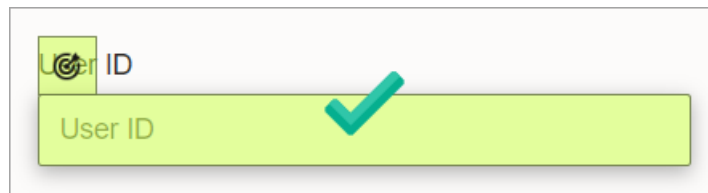
The application that the robot needs to work in opens.

- d. In the application that the robot needs to work in, point to the field where the robot needs to enter its user name, but don't select the field yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Username locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

- f. Repeat the previous steps for the following fields:
  - **Password locator:** Select the field that the robot enters the password into.


- **Submit locator:** Select the button or link that the robot clicks to sign in to the application, such as a **Sign In** or **Submit** button.
6. On the **Input** tab, specify whether to capture any screenshots as part of the action.  
See [Capture Screenshots in Robots](#).
  7. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.  
See [Add Validation to a Robot Action](#).
  8. Click **OK**.
  9. Above the canvas, select **Save**.

## Add an Open Browser Action

The open browser action opens a browser window and enters the URL for an application or website. The open browser action is included in every robot, though you still need to specify the details for the action, including the URL to open.

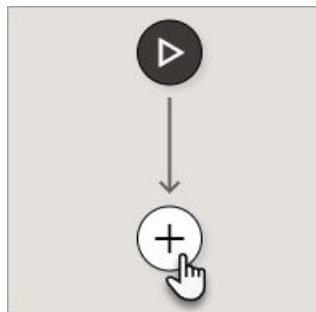
### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. If the robot doesn't contain an open browser action, add it.

Every new robot includes an open browser action, but if you removed it, you can add it back.

- a. Select **Click to insert an action +**.



A menu of available actions appears.

- b. Select **Open Browser**.

An open browser action appears on the canvas, and the Open Browser panel appears.

3. In the panel, enter a **Name** and **Description** for the action.


The Name appears on the action in the canvas and should help you and others understand the goal of the action.

4. On the **Input** tab, enter information about the browser.

- a. Select within the **URL** field.

This field is for the URL that the robot should open. You can hard-code a value, but Oracle recommends specifying a property from the robot connection instead. When you need to update this value in the future, such as to point to a production instance of an application, all you need to do is update the value in the robot connection.

To specify a property from the robot connection, continue following these steps. Or, to review all your options for specifying a value, see [Define the Fields of an Action](#).

- b. In the field, select **More options** **\*\*\***, and then select **Robot connections** .

The Robot connections panel appears.

- c. In the list, expand the entry for a robot connection.

- d. Drag a parameter for the robot connection to the field.

- e. From the **Browser** field, select a browser from the list of options. This list includes all supported browsers, not just the browsers that are installed on a given environment. Make sure you select a browser that is installed on the environment where the robot will run.

- f. If the browser that the robot runs in should run without launching a visible browser on the computer or virtual machine, select **Headless**.

The robot runs the same regardless of your selection. For example, the browser page renders as expected, and the robot captures screenshots as expected. The only difference is that the browser is never visible on the computer. The headless option is useful if the robot works on a computer that a person works on. With a headless browser, the robot doesn't interrupt the person, and the person can't interrupt the robot.

5. On the **Input** tab, if you want to capture a screenshot after the robot opens the browser, select **Full page screenshot after this action**.
6. If you need to switch back to this browser later in the robot: On the **Output** tab, select the variable that stores the index for the browser session.

The index is an assigned number. The index for the first open browser action in the robot is 1, the index for the second browser action in the robot is 2, and so on.

If the entire robot stays in a single internet browser, or if the robot opens one or more other internet browsers and doesn't need to return to this browser, you don't need to enter any values on the Output tab.

 **Tip:**

To learn more about how to switch between browsers, see [Use Case: Switch Browsers](#).

- a. Select within the **Save to** field, and select **Variables** **(x)**.

The Variables panel appears.

- b. If the variable to use doesn't exist yet, create it. See [Create a Variable](#).

- c. From the Available variables list, select a variable, and drag it to the **Save to** field.
7. On the **Post Validate** tab, specify whether to complete any validation after the action  
See [Add Validation to a Robot Action](#).
8. Click **OK**.
9. Above the canvas, select **Save**.

Next, define additional actions if needed. For example, the robot might need to sign in to the application you just opened. See [Add a Login Action](#).

For all available actions, see [Add an Action to a Robot](#).


## Add a Radio Button Action

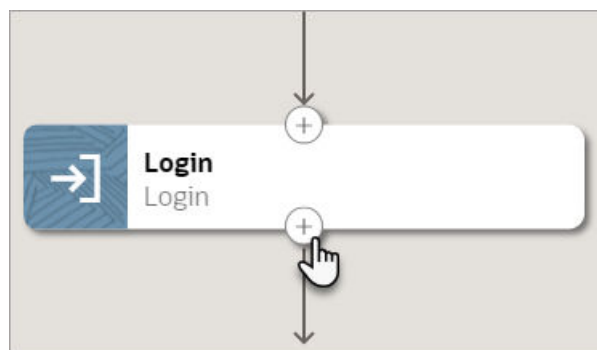
The radio button action interacts with a radio button, such as by selecting it, determining whether any radio buttons in a group are selected, or determining whether a specific radio button is selected.

This action has specific requirements for the page's underlying HTML code. You can use this action only when a UI element element has a tag name of `INPUT` and a type attribute of `radio`.

### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

- b. Select **Radio Button**.

A Radio Button action appears on the canvas, and the Radio Button panel appears.

3. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.

4. From **Operations**, select how the robot interacts with the radio button.

- **Select Radio Button:** Select a radio button.
- **Is Radio Button Selected:** Determine whether any radio buttons in a group are selected.
- **Is Radio Button Set To:** Determine whether a radio button that you specify is selected.

5. On the **Input** tab, specify details for the **Value** field, if it appears.

If the operation that you selected requires you to target an individual radio button, the Value field appears on the Input tab.

- a. In another browser window, open the application that the robot needs to work in.
- b. Right-click the radio button that the robot needs to interact with, and select **Inspect**.
- c. In the HTML code for the page, locate the `id` or the `value` of the radio button.
- d. Type the value for the `id` or `value` into the **Value** field in the robot.

You don't need to format the value in a specific way. For instance, if the `id` is `choice1`, type **choice1**.

6. On the **Input** tab, specify details for the **Group Name** field.

Every operation requires you to select the radio button group.

- a. In another browser window, open the application that the robot needs to work in.  
If you opened the application already, you don't need to open it again.
- b. Right-click one the radio button group that the robot needs to interact with, and select **Inspect**.
- c. In the HTML code for the page, locate the `name` for the radio buttons in the radio button group.
- d. Type the value for the `name` into the **Group Name** field in the robot.


7. On the **Input** tab, specify whether to capture any screenshots as part of the action.

See [Capture Screenshots in Robots](#).

8. If the **Output** tab appears, specify where to save the radio button information that you get from this action. You have the following options:

Some radio button operations create an output.

- Assign the value to a variable. The value is a boolean value, indicating whether a radio button is selected.
  - a. Click within the **Save to** field, and select **Variables** (x).  
The Variables panel appears.
  - b. Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#). The variable must have a type of boolean.
  - c. Select the variable to assign the value to, and drag it to the **Save to** field.
- Assign the value to a property of the output property. The value is a boolean value, indicating whether a radio button is selected.

- a. Click within the **Save to** field, select **More options**, and then select  **Output**.  
The Output panel appears.
  - b. Determine whether the output property that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#). The property must have a type of boolean.
  - c. Select the output property to assign the value to, and drag it to the **Save to** field.
9. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.  
See [Add Validation to a Robot Action](#).
  10. Click **OK**.
  11. Above the canvas, select **Save**.

## Add a Screenshot Action

The screenshot action captures a screenshot of a user interface and sends the file to the integration that ran the robot.


The screenshot action is different from the screenshots that you can capture before and after an action occurs as part of troubleshooting. See [Capture Screenshots in Robots](#).

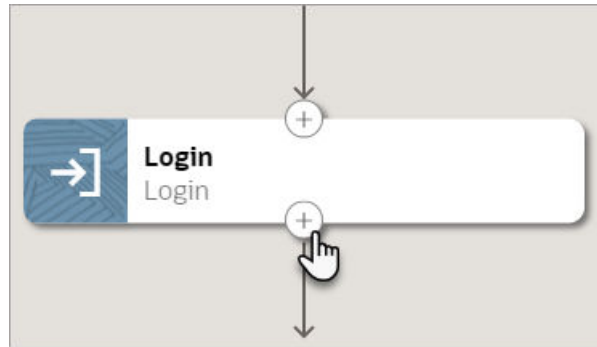
You typically use the screenshot action when the screenshots that are available within an action don't meet your requirements. For instance:

- If an action opens a new tab or browser, use the screenshot action to capture an image of the new tab.
- If you need to send a screenshot to another application, use the screenshot action to capture the image.

### Note:

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

b. Select **Screenshot**.

A Screenshot action appears on the canvas, and the Screenshot panel appears.

3. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.

4. Specify how much of the user interface to capture in the screenshot:

- **Full page:** Capture the application's entire page in the screenshot.
- **Element locator:** Capture only a specific element in the user interface in the screenshot.

If you choose this option, the **Locator** field appears. In this field, specify the element to capture.

a. In another browser window, open the application that the robot needs to work in.

b. In the robot, click within the **Locator** field, and select **Target a page element** .

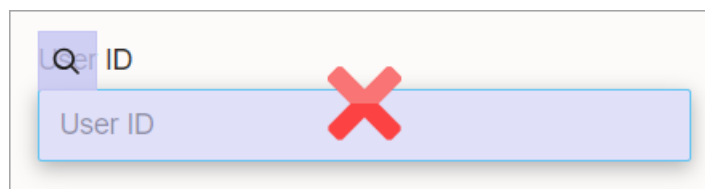
The Target a page element panel appears.

c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

The application that the robot needs to work in opens.

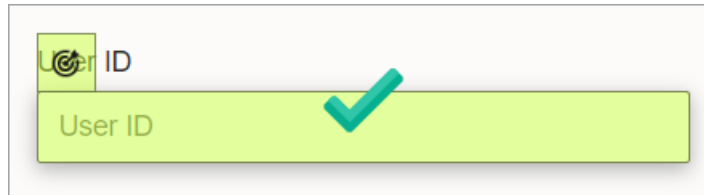
d. In the application that the robot needs to work in, point to the field from which the robot needs to get text, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.





For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.


5. Click **OK**.
6. Above the canvas, select **Save**.

## Add a Switch Browser Action

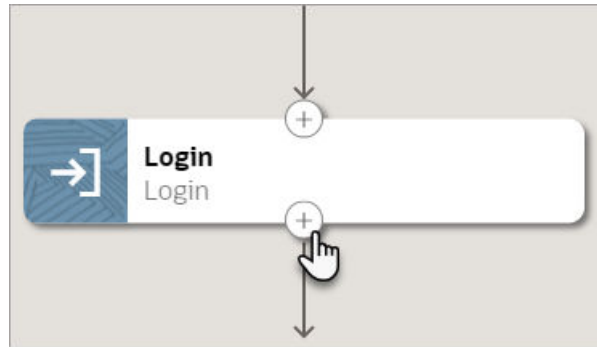
The switch browser action stops working in your current browser window or tab and switches to a different browser window or tab that is already open. You can also use the switch browser action to move past pop-up windows and pages.

 **Note:**

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.
2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

**b.** Select **Switch Browser**.

A Switch Browser action appears on the canvas, and the Switch Browser panel appears.

**3.** In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.

**4.** On the **Input** tab, in the **Browser window** field, specify the index value of the browser window to switch to.

You define the index value using the output for an [open browser action](#). You can select the variable that stores the index value or hard-code a value. The index for the first open browser action in the robot is 1, the index for the second browser action in the robot is 2, and so on.

 **Tip:**

To learn more about how to switch between browsers, see [Use Case: Switch Browsers](#).

**a.** Select within the **Browser window** field, and select **Variables (x)**.

The Variables panel appears.

**b.** If the variable to use doesn't exist yet, create it. See [Create a Variable](#).

**c.** From the Available variables list, select a variable, and drag it to the **Browser window** field.

**5.** On the **Input** tab, if you want to capture a screenshot after the robot switches browsers, select **Full page screenshot after this action**.

**6.** On the **Post Validate** tabs, specify whether to complete any validation after the action.

See [Add Validation to a Robot Action](#).

**7.** Click **OK**.

**8.** Above the canvas, select **Save**.



## Add a Wait Until Element Is Visible Action

The "wait until element is visible" action pauses a robot until a specific element in a user interface is visible.

Specify the maximum time to wait, such as 30 seconds. As soon as the element becomes visible, even if the entire time period hasn't elapsed, the robot continues running. If the element doesn't appear within the time, the robot fails.

## Add a Wait Until Element Is Visible Action Using the Recorder

Consider creating any robot resources that the robot action will use, such as variables or a trigger's input and output. If you choose not to create them in advance, you can pause the recorder and create them as you build.

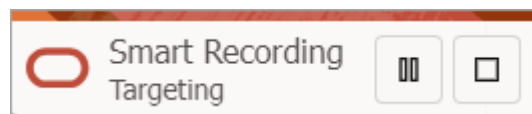
1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Start the recorder.
  - a. On the canvas, select the action that you want to record after.
  - b. On the toolbar, select **Record after the selected action** .
  - c. In the Smart record panel, open the **Select browser tab to target** drop-down, and select the application that you want to work in. You might need to scroll down to find the application.

If the application doesn't appear in the list, close the Smart record panel, open the application in another tab, and start the recorder again.

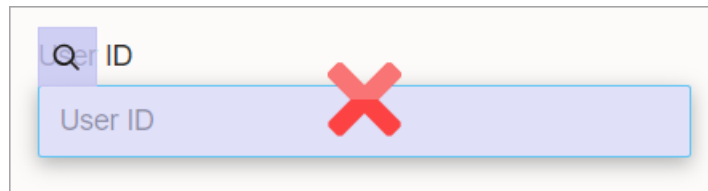
A splash screen appears, and then the RPA Smart Recording panel appears.

- d. Select **Begin Recording**.

The Smart Recording window appears in the lower-left corner of your browser. Additionally, your mouse cursor can now target elements in the user interface.



3. Identify the element in the user interface that the robot needs to interact with.
  - a. Point to the field from which the robot needs to click a UI element, but don't select the element yet.  
For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.






- b. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).


The Smart Recorder panel appears with details about the element you selected.

- c. Review and update the fields as needed.
- **Name:** Enter the name of the target. This text appears on the robot action in the canvas and in the list of targets that are in the robot.  
Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.
  - **Target name:** Review the XML Path Language, or XPath, for the element that you selected. Underscores ( `_` ) appear in place of invalid characters. You can update the value, if you want.
  - **Element:** Review the HTML element that you selected. If you selected the wrong element type, select **Discard** in the panel, and select a different element.
  - **Action:** Select **Wait Until Element Is Visible**.
- d. Select **Save**.
4. Choose the appropriate next step:
- Add another action using the recorder.  
See [Add an Action to a Robot](#).
  - To pause the recorder so you can figure out your next steps, select **Pause**  in the Smart Recording window in the lower-left corner of the browser.
  - To stop the recorder and return to the canvas, select **Stop**  in the Smart Recording window in the lower-left corner of the browser.  
If you close the application that you're recording in, you can still stop the recorder.  
Select **Stop**  in the toolbar of the canvas.
  - To customize the action you just added, such as by creating validation or identifying the screenshots to capture, stop the recording, double-click the action on the canvas, and update the action as needed.

All actions are read-only until you stop the recorder.

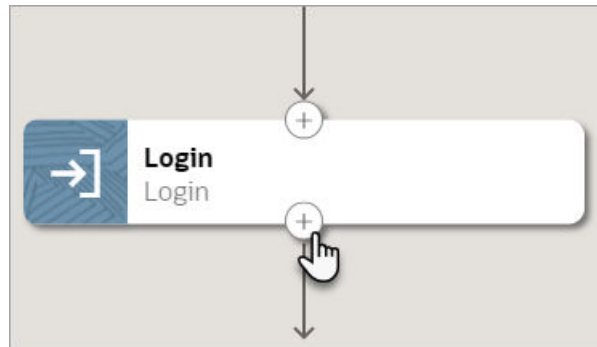
5. Above the canvas, select **Save**.

## Add a Wait Until Element Is Visible Action Using the Low-Code Tools

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.


2. Add the action to the robot.
  - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.

- b. Select **Wait Until Element Is Visible**.

A Wait Until Element Is Visible action appears on the canvas, and the Wait Until Element Is Visible panel appears.
3. In the panel, enter a **Name** and **Description** for the action.

The Name appears on the action in the canvas and should help you and others understand the goal of the action.
  4. On the **Input** tab, specify input details for the action.
    - a. In another browser window, open the application that the robot needs to work in.
    - b. In the robot, click within the **Locator** field, and select **Target a page element** .

The Target a page element panel appears.

    - c. In the Target a page element panel, open the **Select browser tab to target** drop-down, select the application that you want to work in, and select **Go**. You might need to scroll down to find the application.

The application that the robot needs to work in opens.
    - d. In the application that the robot needs to work in, point to the field that the robot needs to confirm is visible, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- e. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the **Locator** field in the robot.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

- f. In the **Timeout** field, enter the maximum amount of time wait in seconds.
- 5. On the **Input** tab, specify whether to capture any screenshots as part of the action.  
See [Capture Screenshots in Robots](#).
- 6. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.  
See [Add Validation to a Robot Action](#).
- 7. Click **OK**.
- 8. Above the canvas, select **Save**.

## Deeper Dive: Settings for Robot Actions

The settings on a robot action allow you to tailor the robot's behavior to meet your goals. Learn more about your options for customizing the behavior of robot actions.

### What Do You Want to Learn About?

| Goal  | Link   |
|---|--|
| Use one of the following placeholder values, rather than hard-coded values, when defining a robot action: <ul style="list-style-type: none"> <li>Robot connection</li> <li>Target</li> <li>Input or output property of a trigger</li> <li>Variable</li> </ul> | <a href="#">Define the Fields of an Action</a>   |
| Choose the screenshots to capture in a robot action   | <a href="#">Capture Screenshots in Robots</a>    |
| Add validation to a robot action  | <a href="#">Add Validation to a Robot Action</a> |

## Define the Fields of an Action

If you're new to designing robots, spend some time familiarizing yourself with your options for defining the fields for an action.




### On This Page










- [How to Define Field Values](#)
- [Examples](#)
- [Additional Options When Defining an Action](#)

### How to Define Field Values

The following table helps you understand all the available options and how to apply them.

Some fields include only some of these options.

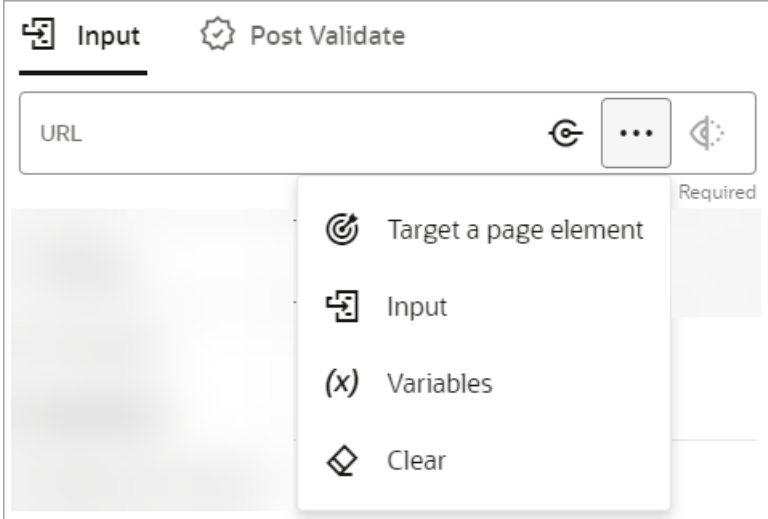
| Goal  | More information  |
|---|---|
| Use a value from a robot connection<br> <b>Robot connections</b> | <ol style="list-style-type: none"> <li>In a robot action, select within the field to define, and select <b>Robot connections</b> . This option is sometimes in the <b>More options</b>  menu. The Robot connections panel appears.</li> <li>In the list, expand the entry for a robot connection.</li> <li>Drag a parameter for the robot connection to the field.</li> </ol> |

| Goal   | More information  |
|--|---|
| <p>Choose an element in a user interface</p> <p> <b>Target a page element</b></p> | <p><b>To select a field using the recorder:</b></p> <ol style="list-style-type: none"> <li data-bbox="643 296 1474 394">1. In a robot action, select within the field to define, and select <b>Target a page element</b> . This option is sometimes in the <b>More options</b> <b>***</b> menu.</li> <li data-bbox="643 415 1474 590">2. In the application that the browser needs to work in, point to the field where the robot needs to enter its user name, but don't select the field yet.<br/><br/>For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.</li> </ol> <div data-bbox="691 604 1398 785" style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;">  </div> <ol style="list-style-type: none"> <li data-bbox="643 814 1474 863">3. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.</li> </ol> <div data-bbox="691 877 1398 1066" style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;">  </div> <div data-bbox="691 1100 1468 1360" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0; background-color: #e0f2f1;"> <p> <b>Tip:</b></p> <p>Your <a href="#">settings</a> determine whether Oracle Integration reuses targets for previously selected UI controls. You can <a href="#">override this setting</a>, if needed. Reusing a target offers benefits. For example, you can <a href="#">update a target</a> one time, and all robots that use the target get the update.</p> </div> |
| <p>Use an input property from the robot trigger</p> <p> <b>Input</b></p>        | <ol style="list-style-type: none"> <li data-bbox="643 1402 1474 1501">1. In a robot action, select within the field to define, and select <b>More options</b> <b>***</b>, and then select <b>Input</b> .<br/>The Input panel appears.</li> <li data-bbox="643 1522 1474 1556">2. From the <b>Properties</b> list, select a property, and drag it to the field.</li> </ol>  |
| <p>Use an output property from the robot trigger</p> <p> <b>Output</b></p>      | <ol style="list-style-type: none"> <li data-bbox="643 1598 1474 1696">1. In a robot action, select within the field to define, and select <b>More options</b> <b>***</b>, and then select <b>Output</b> .<br/>The Output panel appears.</li> <li data-bbox="643 1717 1474 1751">2. From the <b>Properties</b> list, select a property, and drag it to the field.</li> </ol>  |

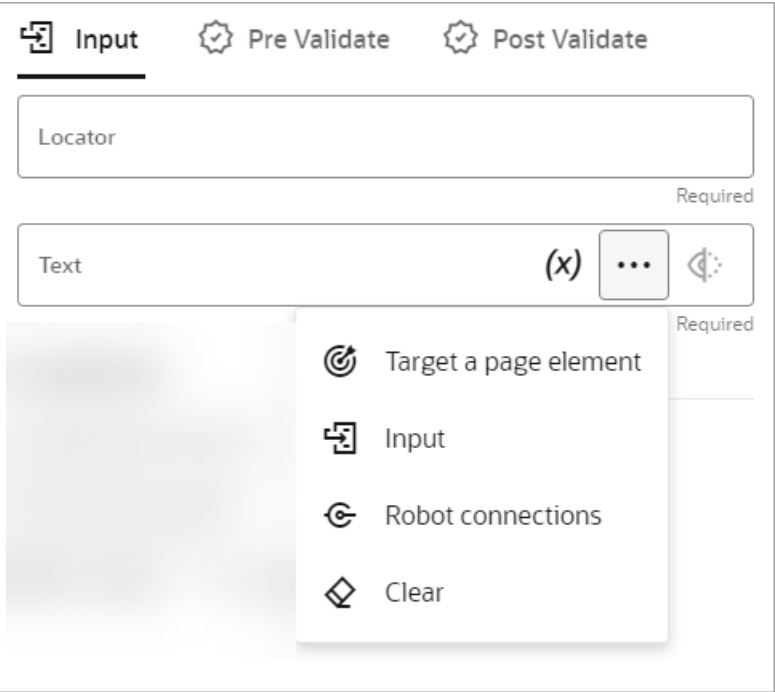


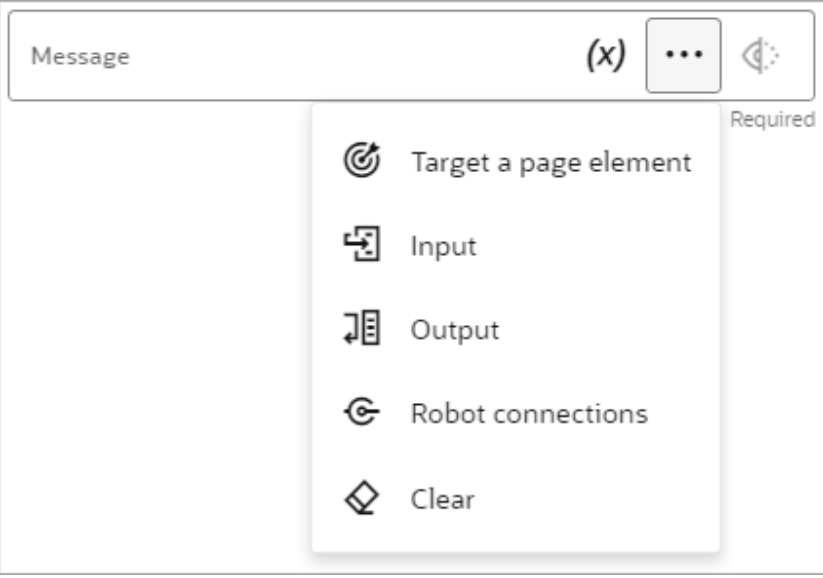
| Goal   | More information   |
|--|--|
| Use a project-level variable<br><b>(x) Variables</b> | <ol style="list-style-type: none"> <li>In a robot action, select within the field to define, and select <b>Variables (x)</b>. This option is sometimes in the <b>More options</b> *** menu. The Variables panel appears.</li> <li>If the variable to use doesn't exist yet, create it. See <a href="#">Create a Variable</a>.</li> <li>From the <b>Available variables</b> list, select a variable, and drag it to the field.</li> </ol> |
| Hard code a value by typing text                     | <p>In general, Oracle recommends using placeholder values rather than hard-coded values. Placeholder values are easy to reuse and update. However, you can hard-code a value if needed when defining a robot action.</p> <ol style="list-style-type: none"> <li>In a robot action, select within the field to define.</li> <li>Enter the value for the field.</li> </ol>   |
| Include an expression in a field                     | <p>If you need to take action on a value, such as by using it to obtain a different value through a calculation, you can include an expression in a field. See <a href="#">Guidelines for Expressions</a>.</p>   |

## Examples

| Action       | Example  |
|--------------|--|
| Open browser | <p>The URL field has offers several options for defining a value.</p>  |

| Action | Example |
|--------|---------|
|--------|---------|

|            |   |
|------------|---|
| Enter text | <p>The Text field offers several options for defining a value.</p>  |
|------------|---|

|     |  |
|-----|--|
| Log | <p>The Message field offers several options for the data to include in the activity stream.</p>  |
|-----|--|

**Additional Options When Defining an Action**

Each action has additional options that are specific to the action. For details, see [Add an Action to a Robot](#).

## Capture Screenshots in Robots

When you add an action to a robot, you have several options for capturing screenshots, including capturing a robot's view before and after completing an action.

### Why Capture Screenshots?

Screenshots are helpful when troubleshooting a failed robot instance.

For example, if a robot's credentials don't have the appropriate access, a robot might not be able to see the field it needs to update. A screenshot of the missing field can help you quickly troubleshoot any issues that the robot encounters.

### Screenshot Options

Most robot actions include the following options for capturing screenshots:



**Screenshots**

Before this action

After this action

Specify screenshot type

Full page     Action target

- **Before this action:** Take a screenshot of the application before the action occurs. Screenshots are typically most helpful when you are debugging and troubleshooting a new robot. A robot that captures screenshots sometimes run a little slower and captures more data than a robot that doesn't capture screenshots.
- **After this action:** Take a screenshot of the application after the action occurs.
- **Specify screenshot type:** Select the scope of the screenshot. To capture the entire application screen, select **Full page**. To capture only the UI element that the action affects, select **Action target**.

### A Screenshot Action Is Also Available

You capture screenshots for robot action to use in troubleshooting. If you need to use a screenshot as part of a robot's or integration's other activities, use the screenshot action. See [Add a Screenshot Action](#)

## Add Validation to a Robot Action

After you provide the input details for a robot action, Oracle Integration creates prevalidation for the UI element(s) that you selected. If needed, you can add more prevalidation and postvalidation. Validation can be for the visibility or enabled state of a UI control or for a specific wait time.

The default prevalidation ensures that the UI control from the input action is visible before proceeding with the action. For example, if the input for a click element action is a Submit button, the validation allows the action to start only after the Submit button is visible. If the action must also wait until the Submit button is enabled, add prevalidation for that requirement.

All the validation conditions must be met for an action to proceed.

1. Open a robot.  
See [Open a Robot](#).
2. On the canvas, find the robot action that requires validation, and double-click it.  
The panel for the action appears.
3. Choose the type of validation to add to the robot action by selecting one of the following tabs:
  - To add validation that occurs before the robot performs an action, select the **Pre Validate** tab.
  - To add validation that occurs after the robot performs an action, select the **Post Validate** tab.
4. If you already created a page state for the action, select it from the **Page state** drop-down list.

You can create a page state on another robot action or from the canvas. See [Create a Page State](#).

Select more page states if needed, and then select **OK**. You're now finished adding validation.

If you haven't create a page state for the action, continue to the next step.


5. Add validation for the robot action.  
First, you create a page state, which is a container for one or more validation checks. Then, you add validation checks to the page state.
  - a. Click **Create +**.
  - b. In the **Name** field, enter a name for the page state, such as **Verify that I'm on the order page**.
  - c. Add a validation condition: Next to Conditions, click **Create +**.
  - d. Fill in the following fields.

| Field    | Description  |
|----------|--|
| Name     | Enter a name for the validation condition, such as <b>Submit button enabled</b> .  |
| Message  | Enter the message that appears when the validation condition isn't met before the timeout period elapses. For example, <b>The Submit button did not become enabled within the specified 30-second timeout period</b> . If the condition times out, the robot fails, and this message appears in the activity stream.   |
| Strategy | Select one of the following options: <ul style="list-style-type: none"> <li>• <b>Element</b>: Add validation that is related to the visibility or enabled state of a UI element.</li> <li>• <b>Fixed Time</b>: Add a predetermined wait period.</li> </ul> <b>Caution:</b> The Fixed Time validation introduces explicit slowness. Consider your requirements carefully before adding this validation, and use it sparingly. |

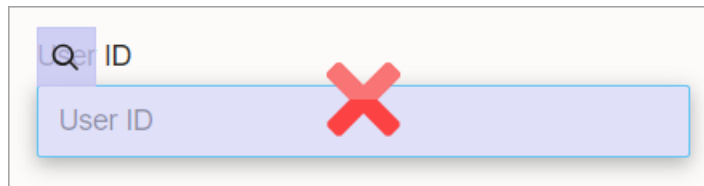
| Field             | Description   |
|-------------------|---|
| State             | (Visible only if you select <b>Element</b> for the <b>Strategy</b> )<br>Select one of the following options: <ul style="list-style-type: none"> <li>• <b>Visible:</b> The validation condition checks whether a UI element appears before the action proceeds.</li> <li>• <b>Enabled:</b> The validation condition checks whether a UI element is clickable before the action proceeds.</li> </ul>  |
| Timeout (seconds) | Enter the timeout period for the validation. <ul style="list-style-type: none"> <li>• If you selected <b>Element</b> for the <b>Strategy:</b> Enter the maximum amount of time to wait for the condition to become true. The action runs as soon as the condition becomes true; this value is simply the maximum amount of time to wait before the robot instance fails.</li> <li>• If you selected <b>Fixed Time</b> for the <b>Strategy:</b> Enter the specific wait time that the robot waits every time it runs.</li> </ul> |

- e. If you selected **Element** for the **Strategy**): Next to Selectors, click **Create +**, and identify the UI control for the validation.

For example, if the validation ensures that a button is visible, you target the button.

- i. In another browser window, open the application that the robot needs to work in.
- ii. In Oracle Integration, click within the **Value** field, and select **Target a page element** .
- iii. In the application that the browser needs to work in, point to the UI control that you're adding validation to, but don't select the UI control yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



- iv. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the Value field in Oracle Integration.

 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

- f. Next to **Create page state**, click **OK**.
6. If needed, repeat the previous steps to add more validation.
- If you need to update a page state or its conditions, see [Update a Page State](#).

## Add Logic to a Robot

Logic helps your robot complete complex tasks.

### What Does the Robot Need to Do?

| Flow control                               | Description                            |
|--|--|
| Loop through a collection of items         | <a href="#">Add a Foreach Loop</a>     |
| Take different actions based on conditions | <a href="#">Add a Switch Condition</a> |
| Stop running                               | <a href="#">Add a Stop</a>             |

## Add a Foreach Loop

When a robot must perform the same actions for multiple items, include the actions in a foreach loop. A foreach loop instructs a robot to iterate through a collection of items, one at a time. Provide the items to iterate upon using a variable collection.

### Use Case

Consider an organization that needs to manually update several invoices every day. Within a robot, they create a foreach loop, which contains the actions for updating the invoices. The input for the robot is a variable collection, which contains the invoices that require updates.

### Prerequisites




A foreach loop requires a collection variable, which is an array of values. In the use case above, the variable contains the invoice numbers that require updates. You can define the variable at any time, including while you're adding the foreach loop.

### Add a Foreach Loop

 **Note:**

You must use the low-code capabilities to add this logic to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.

- a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add the logic to the robot.
    - a. On the canvas, point to an action, and click **+**.  
A menu of available actions appears.
    - b. Select the **Flow control** tab.
    - c. In the list, select **foreach**.  
A foreach action appears on the canvas, and the foreach panel appears.
  3. In the panel, enter a **Name** and **Description** for the action.
  4. In the **Collection** field, specify the data set that the foreach loop iterates on.  
You have the following options:
    - Use the data set from a collection variable.
      - a. Click within the **Collection** field, and select **Variables** .
      - The Variables panel appears.
      - b. Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#). When creating a variable, ensure the following:
        - The variable must be a collection.
        - The properties within the variable must not be collections.
      - c. Select the variable to assign the value to, and drag it to the **Collection** field.
    - Use the data from an output property that is a collection.
      - a. Click within the **Collection** field, select **More options**, and then select  **Input**.  
The Output panel appears.
      - b. Determine whether the input property that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).  
The property must be a collection.
      - c. Select the input property to assign the value to, and drag it to the **Collection** field.
  5. In **Iteration parameter**, enter the name to give to every record in the collection.  
The **Iteration parameter** becomes a variable that you can reference in any action within the foreach loop.  
For example, if you chose an `Invoice` variable for **Collection**, you might enter `Current_Invoice` as the `Iteration` parameter.
  6. Determine whether to select **Continue iteration on error**:
    - When **Continue iteration on error** is selected and an error occurs while a robot instance works on a record, the robot instance records the error, stops performing any additional actions on the record, and moves on to work on the next record. For

instance, if the foreach loop contains a logger action, the robot instance doesn't complete the logging work for any records with errors.

This approach follows a fail-safe methodology.

- When **Continue iteration on error** is not selected and an error occurs while a robot instance works on a record, the robot instance records the error and stops all work on all records.

This approach follows a fail-fast methodology.

7. On the **Pre Validate** and **Post Validate** tabs, specify whether to complete any validation before and after the action.

See [Add Validation to a Robot Action](#).

8. Click **OK**.
9. Above the canvas, select **Save**.

Next, add actions that the robot needs to complete within the foreach loop. For example, you might start by adding the Log action so that you can record the information that is passed into the foreach loop. See [Add a Log Action](#) or [Add an Action to a Robot](#).

## Add a Switch Condition

When the presence or absence of circumstances determines a robot's next step, add a switch condition to a robot. A switch condition evaluates one or more conditions and then takes the appropriate action.

### Sample Use Case

A robot must complete the following tasks:

- Compare the supplier name that is passed into the robot with the supplier name that appears on an invoice.
- Compare the invoice total that is passed into the robot with the total that appears on an invoice.
- Take the appropriate action:
  - If the supplier names and invoice totals match, update the invoice.
  - If the supplier names and invoice totals don't match, do nothing.

Use a switch condition to achieve these goals. A condition in the switch condition uses an expression to determine whether all values match. Include the action to take in the condition. An otherwise condition contains the alternative action--in this case, doing nothing.

For more details about this use case, see [Use Case: Update a Set of Invoices](#).

### Add a Switch Condition




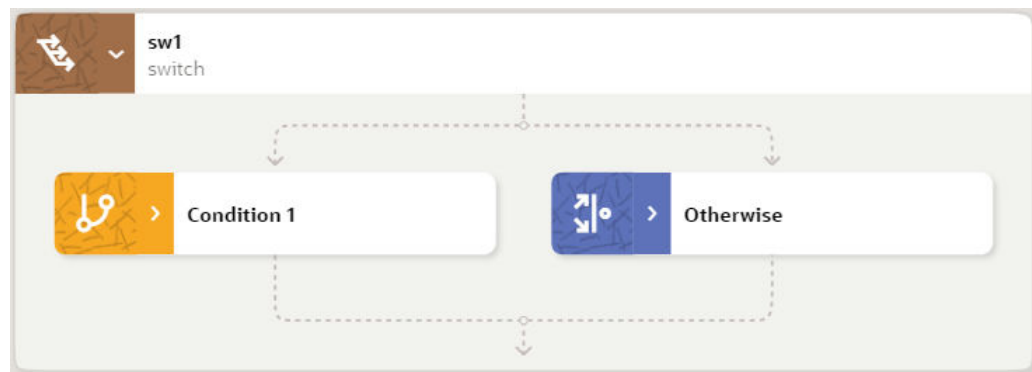
#### Note:



You must use the low-code capabilities to add this logic to a robot. Keep reading for step-by-step instructions.

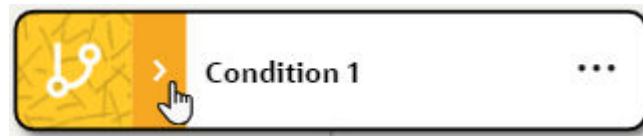
1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.



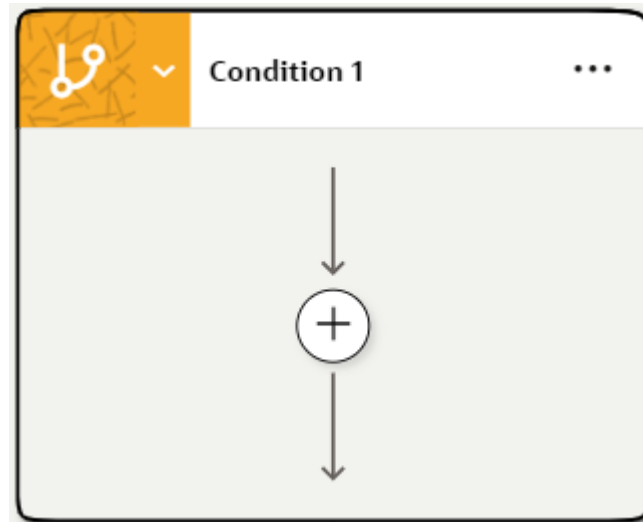
- b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add the logic to the robot.
    - a. On the canvas, point to an action, and click **+**.  
A menu of available actions appears.
    - b. Select the **Flow control** tab.
    - c. In the list, select **switch**.  
A switch condition appears on the canvas, and the Switch condition panel appears.



3. Name the switch condition.
  - a. In the panel, enter a **Name** and **Description** for the action.
  - b. Select **OK**.
4. Define the first condition in the switch condition.  
For example, the condition might include an expression to determine whether an invoice meets specific criteria.
  - a. Within the switch box on the canvas, point to **Condition1**, select **Show node menu** , and then select **Edit**   
The switch condition panel appears.
  - b. Fill in the following fields:
    - **Name:** Enter a name for the condition. The name appears in the box for the condition on the canvas.
    - **Description:** Enter a description for the condition.
    - **Condition:** Enter the expression that the condition must evaluate. If the expression returns `true`, the switch performs the actions that you define in the condition box.  
  
The expression can include variables, inputs, page states, robot connections, and expression language. For help formatting an expression, see the following resources [Guidelines for Expressions](#).
5. Add one or more actions to the first condition.
  - a. Point to the condition, and select the **>** button to expand it.



The condition expands, and a plus sign appears in it.




- b. Define actions or logic that the robot takes if the condition returns `true`.

See [Add an Action to a Robot](#) and [Add Logic to a Robot](#).

6. If needed, add more conditions to the switch.

For example, if the robot should take Path A if one condition is `true`, Path B if another condition is `true`, and Path C if both conditions are `false`, add another condition to define Path B.


- a. Point to the switch condition on the canvas, and select **Insert switch condition** 

- b. Use the previous steps to define the condition and its actions and logic.

7. Define the otherwise condition, and add one or more actions to the condition.

If all the other conditions return `false`, the robot completes the actions in the otherwise box.

- a. Within the switch box on the canvas, point to **Otherwise**, select **Show node menu**

**...**, and then select **Edit** 

The default panel appears.

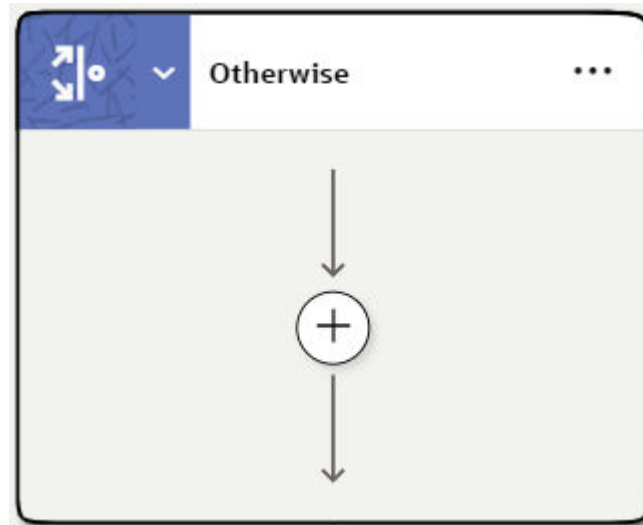
- b. Fill in the following fields:

- **Name:** Enter a name for the otherwise condition. The name appears in the box for the condition on the canvas.
- **Description:** Enter a description for the otherwise condition.

- c. Point to the otherwise condition, and select the **>** button to expand it.



The condition expands, and a plus sign appears in it.



- d. Add actions or logic to the Otherwise box.  
See [Add an Action to a Robot](#) and [Add Logic to a Robot](#).
8. Click **OK**.
9. Above the canvas, select **Save**.

## Add a Stop

Most of the time, you don't need to tell a robot to stop running because it stops when it runs out of actions. However, when you need to add an explicit stop to a robot, add the stop logic, which ends a robot's work.

### Sample Use Case


You typically add a stop within the Otherwise condition of a switch condition. For example, consider a robot that evaluates a condition. If the condition is true, the robot completes the actions in the condition and then completes the remaining actions in the robot. If the condition is false, the robot stops running. Use the stop logic to tell the robot to stop running.

### Add a Stop



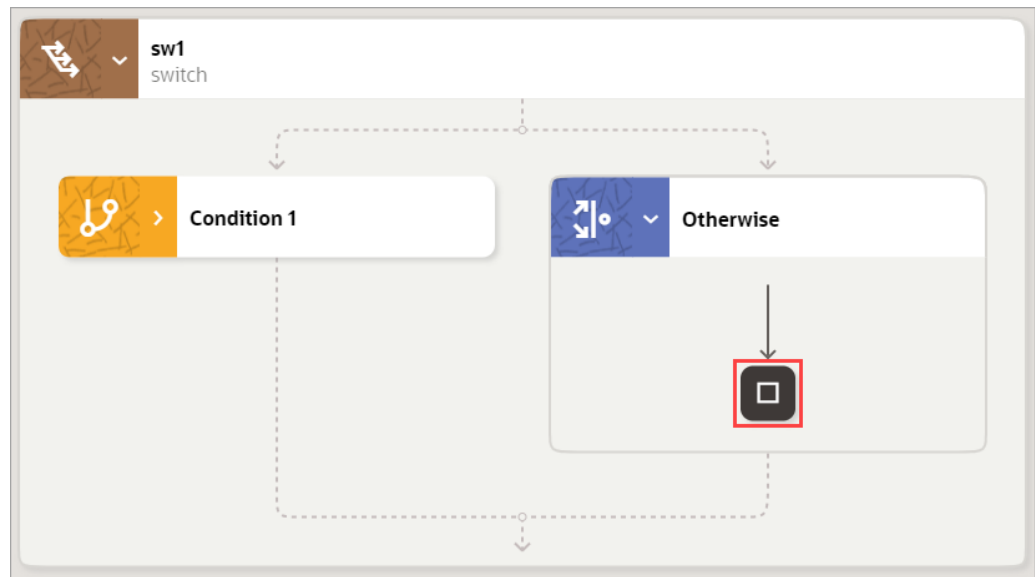
#### Note:

You must use the low-code capabilities to add this logic to a robot. Keep reading for step-by-step instructions.

1. Open the robot to edit.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.

2. Add the logic to the robot.
  - a. On the canvas, point to an action, and click **+**.  
A menu of available actions appears.
  - b. Select the **Flow control** tab.
  - c. In the list, select **stop**.  
A stop action appears on the canvas.



3. Click **OK**.
4. Above the canvas, select **Save**.

## Specify Where a Robot Runs

To test a robot on its environment, you must create an environment pool, add computers to the environment pool, and associate the robot with the environment pool.

### Workflow

1. [Understand the Rules for Environments and Environment Pools](#)
2. [Create an Environment Pool](#)
3. [Add Computers to an Environment Pool](#)
4. [Associate a Robot with an Environment Pool](#)

## Understand the Rules for Environments and Environment Pools

Environments are the computers or virtual machines (VMs) where robots run, and environment pools are collections of these computers. Understand the rules for setting up environments and associating robots with them.

| Rule   | More information   |
|--|--|
| To create an environment, install the robot agent        | A computer or VM becomes an environment that can be added to an environment pool when you install the robot agent on the computer or VM. Start here and work your way through the steps until the robot agent is running on the environment: <a href="#">Meet the Robot Agent's Requirements</a> . |
| Meet a robot's system requirements                       | Every environment in an environment pool should contain everything that a robot needs, such as the correct internet browser.   |
| Robots in the same project can share an environment pool | You can associate multiple robots with one environment pool as long as all the robots are in the same project.<br>You can't associate robots from multiple projects with the same environment pool.  |
| An environment can be part of only one environment pool  | To reduce resource conflicts, you can add an environment to only one environment pool.   |

## Create an Environment Pool

An environment pool is a computer or set of computers that specific robots can run on.


### Prerequisites:

- Install the robot agent on every environment that must be part of the robot's environment pool.

Start here and work your way through the steps until the robot agent is running on the environment: [Meet the Robot Agent's Requirements](#).

- Create the robot that must run on the environment pool.  
See [Create a Robot](#).


### Create an Environment Pool:

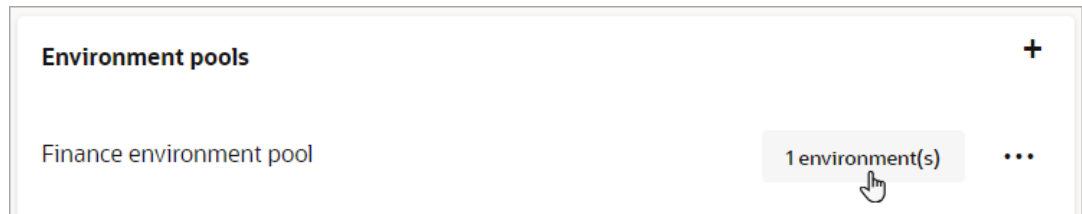
1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. In the **Environment Pools** box, select **Add** or **+**.  
**Add** appears if you have no environment pools. The **+** button appears if you have one or more environment pools.  
The Create environment pool panel appears.
4. Fill in the fields, and select **Create**.

## Add Computers to an Environment Pool

Group related virtual machines into environment pools in a way that makes sense for your organization.


1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.

2. In the left toolbar, select **Robot** .
3. In the **Environment Pools** box, point to the environment pool you just created, select **...**, and select **Add environment**.  
The Add environment panel appears.
4. From the list of options, select one or more environments to add to the environment pool.  
An environment appears in the list only if the robot agent has successfully run on it.
5. Select **Add**.  
The Environment pools box lists the number of environments that are associated with the pool in a button. To view the environments, click the button.



## Associate a Robot with an Environment Pool

To run a robot on a real-world environment, you must first associate the robot with an environment pool.

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. In the **Robots** box, point to a robot, select **...**, and select **Add environment pool**.  
The Add environment pool panel appears.
4. Select the environment pool where the robot will run.
5. Select **Add**.

You now have an environment pool where a robot can run, and the robot is associated with the environment. If the robot doesn't contain any errors, its state changes from Draft to Configured. To find and fix errors, see [Fix a Robot's Errors](#).

Next, activate the robot. See [Activate a Robot](#).

## Design an Integration That Calls a Robot

A robot runs only when an integration calls it. Therefore, you must design an integration to call each robot that you build. An integration developer can design the integration at any time during the development process of the robot.

Before you start designing an integration, make sure that the robot has been created and that its trigger has been defined, including its input and output properties. These properties represent the contract that the robot makes with the integration. You can typically create a robot and define its trigger in just a minute or two. After, an integration developer can start designing the integration that calls the robot. This flexibility allows teams to split work

according to their availability, and the integration and robot developers don't block the other person's work.

A robot developer can fully build and test a robot, even if the integration doesn't exist yet.




1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. Create an integration.
  - a. In the **Integrations** box, click **Add** (if no integrations have been designed) or **+** (if one or more integrations have been designed).

The Add integration panel appears.

- b. Select **Create**.
- c. Select the type of integration to design.

For details about each integration pattern, see *Understand Integration Patterns in Using Integrations in Oracle Integration 3*.
- d. Enter a name for the integration and provide additional descriptive information, if needed.
- e. Select **Create**.

For detailed step-by-step instructions, see *Create an Integration in Using Integrations in Oracle Integration 3*.

3. Call the robot from the integration.
  - a. Add a robot action to the integration using one of the following ways:
    - On the right side of the canvas, click **Actions**  and drag the **Robot flow** action to the appropriate location.
    - Click  at the location where you want to add the for-each action, then select **Robot flow**.
  - b. Point to the **Robot process automation** action, select **...**, and then select **Edit** .

The Configure Basic Info panel appears.
  - c. Fill in the following fields:
    - **What do you want to call your endpoint?:** Enter descriptive text for the robot, such as the robot's name or information about its business process.
    - **What does this endpoint do?:** Provide a brief description of the robot.
  - d. Select **Continue**.
  - e. From the **Flow Name** drop-down, select the robot that the integration calls, including the appropriate version.
  - f. Select **Continue**.
  - g. Review the summary, and select **Finish**.

A Map object appears before the Robot process automation action.
4. Pass information from the integration to the robot.
  - a. Point to the **Map** action that is before the **Robot process automation** action, select **...**, and select **Edit**.

The mapper opens.

- b. In the Sources list on the left, expand the tree until you find the data element that you need to pass to the robot.
- c. In the Target list on the right, expand the tree until you find the input that needs to receive the data.
- d. Drag the source element to the target element.

For more help working in the mapper, see [About Mapping Data Between Applications and Map Data in \*Using the Oracle Mapper with Oracle Integration 3\*](#).

- e. If you need to pass more information to the robot, map additional elements.
- f. After you finish mapping elements, test your mappings.

See [Test Your Mappings in \*Using the Oracle Mapper with Oracle Integration 3\*](#).

- g. Select **Go back** < to return to the canvas.

## Create and Update a Robot Resource

You can create variables, a trigger's input and output, page states, targets, and data types either before build your robot or as you build your robot--it's up to you. If you want to build your robot from start to finish without pausing, create the resources before you build your robot.

You can update these resources at any time from the canvas.

Robot connections are different from other resources. You must create them before you build, and you create and manage them outside a robot. See [Create Connections to Applications](#).

### Learn About Robot Resources

#### [Alternatives to Hard Coding Data](#)

#### What Do You Want to Create or Update?

- [Work with Variables](#)
- [Work with a Trigger's Input and Output](#)
- [Work with Page States](#)
- [Work with Targets](#)
- [Work with Data Types](#)
- [Work with Robot Connections and Robot Connection Types](#)

## Alternatives to Hard Coding Data

Hard coding values in your robot can make updates tedious and time consuming. To reduce your maintenance costs, Oracle Integration offers numerous alternatives to hard coding data.

You can use many of these robot resources in a robot action. See [Define the Fields of an Action](#).

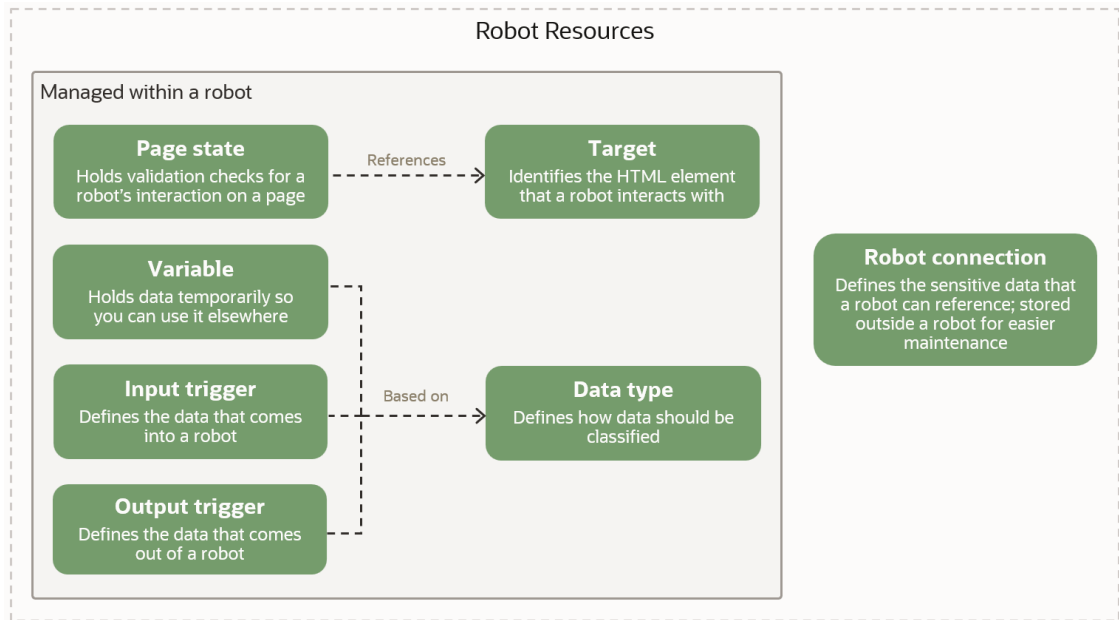
### On This Page

- [Illustration](#)
- [Overview](#)









- [Variables](#)
- [Triggers](#)
- [Page States](#)
- [Targets](#)
- [Data Types](#)
- [Robot Connections](#)

### Illustration



### Overview

| Option   | Why they're helpful  | Example   |
|--|--|---|
| (x) Variables  | A variable holds data temporarily for a robot so that you can use the data somewhere else.<br>Learn more: <a href="#">Variables</a>  | Consider the get text action, which obtains text from a field. You can hold the value in a variable. Additionally, if the text value is an output for the robot, you can use the variable as an output property.  |
|  Input  | An input defines the data that comes into a robot, typically when the robot starts running.<br>Learn more: <a href="#">Triggers</a>  | Consider a robot that needs to update a purchase order. An integration obtains the purchase order number and passes it to the robot as an input property. You dynamically define the input properties in a map action in an integration. You can also define these values when you test a robot outside an integration. |
|  Output | An output defines the data that comes out of a robot. You can dynamically pass the data elsewhere, such as to another robot, integration, or action in an integration.<br>Learn more: <a href="#">Triggers</a> | Consider a robot that obtains the total value of a purchase order. The robot can pass the total value into an output, which an integration can use to take further action.  |

| Option  | Why they're helpful  | Example   |
|---|--|---|
|  Page states         | A page state is a container for one or more validation checks. A validation check occurs before or after an action in a robot.<br>Learn more: <a href="#">Page States</a>  | Consider an application that experiences latency issues during periods of high usage. Without validation checks, a robot that runs in the application could fail if the robot attempts to complete a task before the required fields and buttons are visible. Validation checks ensure that all fields and buttons are visible and enabled on a page before a robot attempts to complete its task. These checks increase the likelihood of a robot completing its tasks successfully. |
|  Targets             | A target identifies the HTML element that a robot interacts with. Every target has two components: a user-friendly name, and the hard-coded XML path language, or XPath, for the element.<br>Learn more: <a href="#">Targets</a>   | Targets help with maintenance. Consider a robot that interacts with the same HTML element multiple times. Oracle Integration creates a target for the element and can reuse the target for each interaction. Therefore, if the HTML for a page changes, all you need to do is update the target one time, and all of the robot's interactions with the element are updated.   |
|  Data types          | A data type defines how data should be classified.<br>A data type is the basis of every variable, output property, and input property. You can use one of the default data types provided by Oracle or a custom data type that you create.<br>Learn more: <a href="#">Data Types</a> | Consider a robot that must update a set of invoices. You can create a data type that defines the components of invoice data, such as invoice number, invoice amount, supplier name, and due amount. Next, you can create an input of the data type, and use the trigger to pass this information into the robot.  |
|  Robot connections | A robot connection lets you store sensitive data, such as sign-in credentials, outside a robot for easier maintenance.<br>Learn more: <a href="#">Robot Connections</a>  | In order for a robot to sign in to an application, you must provide a user name and password. However, an administrator needs easy access to this information so they can update the password according to your organization's security policies. Thanks to a robot connection, this sensitive information lives outside a robot, so you can update it easily outside the lifecycle of the robot.   |

## Variables

| Area                              | More information   |
|-----------------------------------|--|
| <b>How to define the values</b>   | A variable stores one value or a set of values that come from another place. You typically use a variable to hold the output value of a robot action. You cannot hard code values for a variable.  |
| <b>Where to define the values</b> | Create a variable when you <a href="#">add an action to a robot</a> , or <a href="#">at any time on the canvas</a> .   |
| <b>How to use the values</b>      | If you use a variable to hold the output value of a <a href="#">robot action</a> , you can use the variable however you need to. For instance, the variable might be the input for another robot action, a part of an expression, or the output for a robot. |

## Triggers

| Area                              | More information  |
|-----------------------------------|---|
| <b>How to define the values</b>   | Inputs and outputs contain one or more properties. Define the properties, including specifying their data types and whether they hold a single value or an array of values, when you create the trigger.<br><br>You don't define a value for an input or output because a trigger is a container for its properties. Additionally, you can't hard code values for the properties. Instead, you dynamically pass in values for each input and output property. |
| <b>Where to define the values</b> | Create an input or output in a robot, either when you <a href="#">define the trigger of a robot</a> , or <a href="#">at any time on the canvas</a> .  |
| <b>How to use the values</b>      | Use an input or output property when you <a href="#">define a robot action</a> .  |

## Page States

| Area                              | More information  |
|-----------------------------------|---|
| <b>How to define the values</b>   | Page states contain one or more validation checks for a robot action. A validation check uses a target to reference a specific UI control, such as waiting until a button is visible on a page. |
| <b>Where to define the values</b> | Create an input or output in a robot, either when you <a href="#">define the trigger of a robot</a> , or <a href="#">at any time on the canvas</a> .  |
| <b>How to use the values</b>      | Select a page state for a robot action when you <a href="#">define a robot action</a> .   |

## Targets

| Area                              | More information  |
|-----------------------------------|---|
| <b>How to define the values</b>   | Oracle Integration creates targets for you when you select an element for the robot to interact with while building a robot, either using the recorder or the low-code tools.   |
| <b>Where to define the values</b> | Create a target when you <a href="#">add an action to a robot</a> , or <a href="#">at any time on the canvas</a> .  |
| <b>How to use the values</b>      | Your <a href="#">settings</a> determine whether Oracle Integration reuses targets for previously selected UI controls. You can <a href="#">override this setting</a> , if needed. Reusing a target offers benefits. For example, you can <a href="#">update a target</a> one time, and all robots that use the target get the update. |

## Data Types

| Area                              | More information   |
|-----------------------------------|--|
| <b>How to define the values</b>   | A data type can contain one or more properties.<br><br>Oracle Integration provides several predefined data types, including string, boolean, and number.<br><br>You can define additional data types as needed, including defining their properties. |
| <b>Where to define the values</b> | Create a data type <a href="#">at any time on the canvas</a> .   |
| <b>How to use the values</b>      | Select a variable as the basis when you <a href="#">create a variable</a> or <a href="#">create a trigger's input or output</a> .  |

## Robot Connections

| Area                              | More information  |
|-----------------------------------|---|
| <b>How to define the values</b>   | A robot connection contains one or more parameters. Define hard-coded values for the parameters when you create the robot connection. You cannot pass in values to a parameter.<br>Additionally, a robot connection and its parameters live outside a robot, so you can update the parameter values without updating a robot. |
| <b>Where to define the values</b> | <a href="#">Create a robot connection</a> in a project.   |
| <b>How to use the values</b>      | Use the value of a parameter when you <a href="#">define a robot action</a> .   |

## Work with Variables


Create and update variables on the canvas. You can also create variables from within a robot action.

To learn more about variables and other robot resources, see [Alternatives to Hard Coding Data](#).


## Create a Variable

A variable holds data temporarily for a robot so that you can use the data somewhere else. You can add a variable on the canvas or from within a robot action.

If you plan to add actions using the recorder, consider creating the variables that you need on the canvas, before you start building the robot. That way, you won't have to pause the recorder to create the variables.


- Determine where to start:
  - If you're not actively building a robot, start on the canvas.
    - In the navigation pane, select **Projects**.
    - Select the project name.
    - In the left toolbar, select **Robot** .
    - In the **Robots** box, select the robot to open.  
The canvas appears.
    - In the toolbar, select **Variables** (X).  
The Variables panel appears.
  - If you need to add a variable while you're in the middle of adding actions to a robot, work within a robot action.
    - Add an action to a robot.  
See [Add an Action to a Robot](#).
    - While the panel for the robot action is open, select within any field for which you can provide a value, and select **Variables** (X).  
This option is sometimes in the **More options** **...** menu.

The Variables panel appears.



2. Select **Create**  .  
A new variable is added to the list of variables, and the Variable panel appears.
3. Fill in the following fields:
  - **Name:** Enter a name for the variable.
  - **Type:** Select the type on which to base the variable. If the type that you require doesn't appear, you can create it. See [Create a Data Type](#).
  - **Description:** Enter a description for the data type.
  - **Collection:** Select this checkbox if the variable needs to store an array of values. If you leave this checkbox deselected, the variable can store only a single value. For example, if you create a variable to store a value that a robot obtains, and the robot needs to iterate over several records, you must select **Collection** so that the variable can store a value from each record.
4. Click **OK**.
5. Above the canvas, select **Save**.

## Update a Variable

You can update a variable at any time from the canvas, without having to open an action that uses the variable.

1. Open the robot whose variable needs to change.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.

2. On the toolbar, select **Variables**  .  
The Variables panel appears.
3. Select a variable in the list, and select **Edit** .
4. Update the fields as necessary, and select **OK**.
5. Above the canvas, select **Save**.

## Work with a Trigger's Input and Output

Create and update a robot trigger, including the trigger's input and output, on the canvas.

To learn more about triggers and other robot resources, see [Alternatives to Hard Coding Data](#).


## Create a Trigger's Input or Output


An input defines the data that comes into a robot, typically when the robot starts running. An output defines the data that comes out of a robot. You can dynamically pass the data


elsewhere, such as to another robot, integration, or action in an integration. You define the trigger for a robot, including its input and output, on the robot canvas.

You can also define the output when you add the get text action to a robot using the low-code tools. See [Add a Get Text Action](#).

If you plan to add actions using the recorder, consider creating any triggers that you need in advance, so that you don't have to pause the recorder to create the triggers.


1. Open a robot.
    - a. In the navigation pane, select **Projects**.
    - b. Select the project name.
    - c. In the left toolbar, select **Robot** .
    - d. In the **Robots** box, select the robot to open.


The canvas appears.
    - e. In the toolbar, select **Trigger** .

The Trigger panel appears.
2. Select one of the following tabs:
  - **Input:** Define a trigger's input.
  - **Output:** Define a trigger's output.
3. Define the trigger.
  - a. Select **Add** .
  - b. Fill in the fields.
    - **Name:** Enter the name of the property of the trigger. For example, an input named `Person` might have properties of Name, Age, and so on.
    - **Type:** Select the type on which to base the variable. If the type that you require doesn't appear, you can create it. See [Create a Data Type](#).
    - **Collection:** Select this checkbox if the trigger needs to store an array of values. If you leave this checkbox deselected, the trigger can store only a single value.
  - c. Add additional properties as needed.
4. Click **OK**.
5. Above the canvas, select **Save**.

## Update a Trigger's Input or Output

You can update a trigger at any time from the canvas, without having to open the actions that use the trigger.

1. Open the robot whose input or output needs to change.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .

- d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. In the toolbar, select **Trigger**  .  
The Trigger panel appears.
3. Select one of the following tabs:
  - **Input:** Update a trigger's input.
  - **Output:** Update a trigger's output.
4. Expand the property to edit.
5. Update the fields as necessary, and select **OK**.
6. Above the canvas, select **Save**.

## Work with Page States

You create and update a page state on the canvas and within a robot.

If you prefer to prebuild as many components as you can before you start building a robot, you might prefer creating page states from the canvas and then selecting them as appropriate for each robot action.

To learn more about page states and other robot resources, see [Alternatives to Hard Coding Data](#).




## Create a Page State

A page state is a container for one or more validation checks. A validation check occurs before or after an action in a robot. You create a page state, including its validation checks, on the canvas.


You can also create and update a page state when you add an action to a robot. See [Add Validation to a Robot Action](#).

When you create a page state on the canvas, Oracle Integration doesn't apply its validation checks to a robot action. Later, when you select the page state for a robot action, you specify whether the validation must occur before or after the action occurs.

All the validation conditions must be met for an action to proceed.

1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. In the toolbar, select **Page states**  .  
The Page states panel appears.
3. Select **Create**  .


The Page state panel appears.

4. In the **Name** field, enter a name for the page state, such as **Verify that I'm on the order page**.
5. Add one or more conditions to the page state.
  - a. Next to Conditions, select **Create** .
  - b. Fill in the following fields.

| Field             | Description   |
|-------------------|---|
| Name              | Enter a name for the validation condition, such as <b>Submit button enabled</b> .   |
| Message           | Enter the message that appears when the validation condition isn't met before the timeout period elapses. For example, <b>The Submit button did not become enabled within the specified 30-second timeout period</b> . If the condition times out, the robot fails, and this message appears in the activity stream.  |
| Strategy          | Select one of the following options: <ul style="list-style-type: none"> <li>• <b>Element</b>: Add validation that is related to the visibility or enabled state of a UI element.</li> <li>• <b>Fixed Time</b>: Add a predetermined wait period.</li> </ul> <b>Caution:</b> The Fixed Time validation introduces explicit slowness. Consider your requirements carefully before adding this validation, and use it sparingly.  |
| State             | (Visible only if you select <b>Element</b> for the <b>Strategy</b> )<br>Select one of the following options: <ul style="list-style-type: none"> <li>• <b>Visible</b>: The validation condition checks whether a UI element appears before the action proceeds.</li> <li>• <b>Enabled</b>: The validation condition checks whether a UI element is clickable before the action proceeds.</li> </ul>  |
| Timeout (seconds) | Enter the timeout period for the validation. <ul style="list-style-type: none"> <li>• If you selected <b>Element</b> for the <b>Strategy</b>: Enter the maximum amount of time to wait for the condition to become true. The action runs as soon as the condition becomes true; this value is simply the maximum amount of time to wait before the robot instance fails.</li> <li>• If you selected <b>Fixed Time</b> for the <b>Strategy</b>: Enter the specific wait time that the robot waits every time it runs.</li> </ul> |

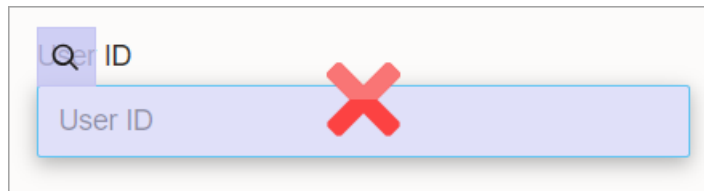
- c. If you selected **Element** for the **Strategy**): Next to Selectors, click **Create +**, and identify the UI control for the validation.

For example, if the validation ensures that a button is visible, you target the button.

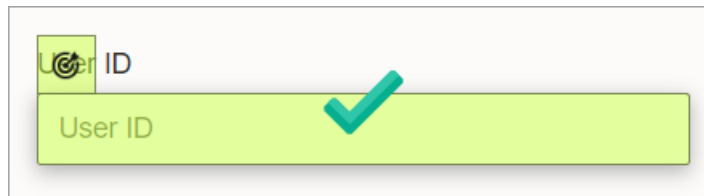
- i. In another browser window, open the application that the robot needs to work in.
- ii. In the Page state panel, click within the **Value** field, and select **Target a page element** .
- iii. In the application that the browser needs to work in, point to the UI control that you're adding validation to, but don't select the UI control yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.





- iv. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).

The recorder enters a value in the Value field in the Page state panel.





 **Tip:**

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override this setting](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all robots that use the target get the update.

- d. Select **OK** in the Page state panel, and then select **OK** in the Page states panel.
  - e. If needed, repeat the previous steps to add more page states or validation conditions.
6. Above the canvas, select **Save**.

## Update a Page State

You can update a page state at any time from the canvas, without having to open the robot actions that use its validation checks.

1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. In the toolbar, select **Page states** .  
The Page states panel appears.
3. Select a page state from the list, and select **Edit** .  
The Page state panel appears.
4. Select a condition from the list, and select **Edit** .

5. Update the fields as necessary, and select **OK**.
6. Above the canvas, select **Save**.

## Work with Targets


Create and update targets on the canvas.




To learn more about targets and other robot resources, see [Alternatives to Hard Coding Data](#).

## Create a Target

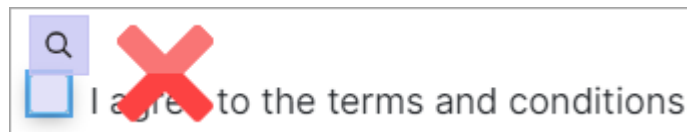
A target identifies the HTML element that a robot interacts with. Every target has two components: a user-friendly name, and the hard-coded XML path language, or XPath, for the element. You can create a target on the canvas, outside of a robot.

You typically create targets while you're adding an action to a robot. See [Add an Action to a Robot](#). Creating a target this way is fast and easy, so you don't save much time by creating targets outside of a robot.

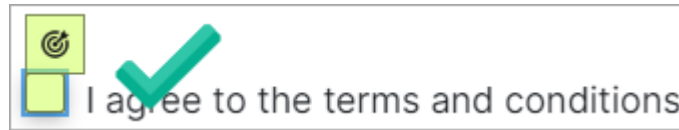
1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.

The canvas appears.
2. In the toolbar, select **Targets** .
3. Select **Create** .
4. In another browser window, open the application that the robot needs to work in.
5. Click within the **Locator** field, and select **Target a page element** .
6. In the application that the robot needs to work in, point to the UI element that a robot needs to interact with, but don't select the element yet.

For example, do not select the UI element while the magnifying lens icon appears and the field is shaded purple. The recorder is still collecting information about the UI element.



7. After the icon changes to a target, the shading turns green, and the mouse icon changes to a hand, select the UI element.



For more tips, see [Quick Start for Building Robots](#).


The recorder enters a value in the **Locator** field.

8. Click **OK**.
9. Above the canvas, select **Save**.


## Update a Target

Sometimes, the fields and buttons that a robot interacts with change. A central list of all targets are right at your fingertips for stress-free maintenance. You don't even have to open the action that references a target.

1. Open the robot whose target needs to change.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.

- c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.


The canvas appears.

2. In the horizontal toolbar on the right, select **Targets** .

The Targets panel appears. The Usages value indicates the number of times the value is used in the robot.

3. Optional: View where the target is used.
      - a. In the list, locate the target to update.
      - b. Select the **Usages** link to its right.

The Target usages panel appears.

- c. Review the actions that use the value.
  4. Update the target.
    - a. Above the list of targets, select **Edit** .

The Target panel appears.

- b. Optional: Update the fields as needed.
  5. Click **OK**.
  6. Above the canvas, select **Save**.

## Work with Data Types



Create and update data types on the canvas.


To learn more about data types and other robot resources, see [Alternatives to Hard Coding Data](#).


## Create a Data Type

A data type defines how data should be classified. Oracle Integration provides several simple data types, including string, boolean, and number. You can create additional data types if needed.

You base variables and a trigger's input and output on data types. For example, consider a variable that must hold several related pieces of data, such as an invoice number, invoice amount, supplier name, and amount due. You can create a data type that contains one property for each piece of data, and then create a variable of the data type. For more information on this use case, see [Use Case: Update a Set of Invoices](#).

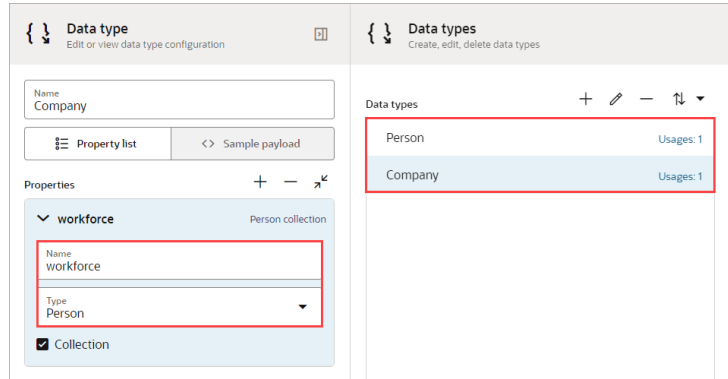
1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. In the toolbar, select **Data types** .
 

The Data types panel appears.
3. Select **Create** .
 

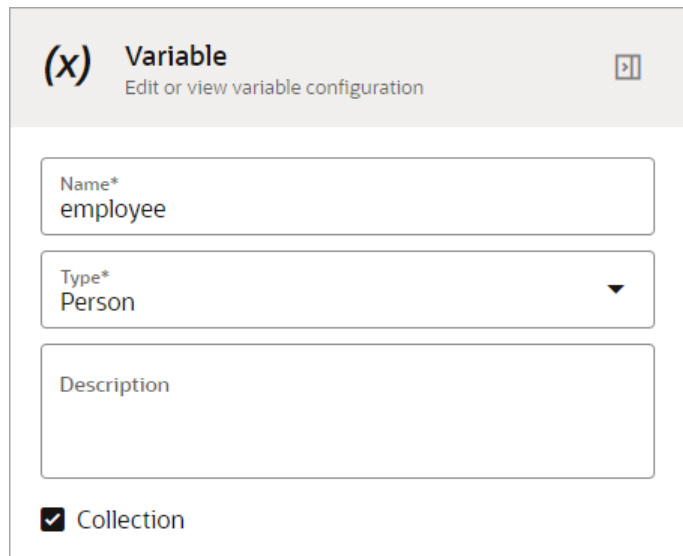
The Data type panel appears.
4. Enter the name of the data type, such as `InvoiceType`.
5. Define the data type using one of the following options:
  - Define the properties manually, one at a time.
    - a. Make sure that the **Property list** tab is selected.
    - b. Next to Properties, select **Add** .
    - c. Fill in the following fields:

| Field | Description  |
|-------|--|
| Name  | Enter the name of the property, such as <code>InvoiceNumber</code> .   |
| Type  | Select the data type of property. You can base a property on the simple data types that Oracle Integration provides or on a data type that you create. |

| Field      | Description  |
|------------|--|
| Collection | <p>Select this checkbox if the property must store an array of values. If you leave this checkbox deselected, the data type can store only a single value.</p> <p>For example, consider a data type of <code>Company</code>, which represents a company, and a data type of <code>Person</code>, which represents an employee.</p> <p>One of the properties of the <code>Company</code> data type is <code>workforce</code>, which is a collection of <code>Person</code> types.</p> |



You can create a `employee` variable that is of the data type `Person`. If you want this data type to be an array of persons, mark it as a collection.



- Upload a JSON file that defines the properties.
  - a. Select the **Sample payload** tab.
  - b. Select within the **Drag and Drop** area, navigate to the location of the JSON file, and double-click the file.

or

Paste the JSON text into the **Enter a JSON sample payload** area.

For example, the following JSON text defines two properties, `Green Corp.` and `employees`. The `employees` property is of type string and is a collection.



```
{
  "name": "Green Corp.",
  "employees": [
    "Horst", "Sandhya"
  ]
}
```


6. In the Data type panel, where you defined the properties, make sure that the tab where you defined the properties--either **Property list** or **Sample payload**--is selected.
7. Click **OK**.
8. Above the canvas, select **Save**.

## Update a Data Type

You can update a data type at any time from the canvas. If you base a variable or a trigger's input or output on a data type, be aware of how updates affect your robot resources.

| Update  | How the update affects a robot  |
|---|---|
| Deleting a property, or changing the name of a property | If an expression or field references the property, an error occurs the next time you validate the robot, and you must correct the error. Validation happens automatically when you save, and you can also explicitly validate a robot. See <a href="#">Validate a Robot</a> . |
| Adding a property                                       | You can add as many properties as you want. Adding a property doesn't impact any existing robot resources or expressions.   |

1. Open the robot whose data type needs to change.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. In the toolbar, select **Data types** .
 

The Data types panel appears.
3. Update the data type.
  - a. Above the list of data types, select **Edit** .  
The Data type panel appears.
  - b. Optional: Update the fields as needed.
4. Click **OK**.
5. Above the canvas, select **Save**.

## Work with Robot Connections and Robot Connection Types


Update robot connections and robot connection types within a project. You don't need to open a robot to update these resources.

To learn more about robot connections and other robot resources, see [Alternatives to Hard Coding Data](#).

### Update a Robot Connection

A robot connection lets you store sensitive data, such as sign-in credentials, outside a robot for easier maintenance. You can update a robot connection at any time without having to open or update a robot. The next time a robot that uses the robot connection runs, the robot instance uses the updated values.


You typically create robot connections before you create a robot, and the robot connection contains details for a test or UAT environment. See [Create Connections to Applications](#). Then, after you're ready to deploy a robot to production, update the robot connection so that it points to a production environment.

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. In the **Robot Connections** box, find the robot connection to update.
4. To the right of the robot connection, select **...**, and then select **Edit**.  
The Edit robot connection panel appears.
5. Update the fields as necessary, and select **Update**.

For example, if you're ready to deploy a robot to production, update the robot connection so that it contains the production URL and credentials.

### Update a Robot Connection Type

You can update a robot connection type at any time. However, if a connection is already based on the robot connection type, the change might make the robot connection no longer valid.

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. In the **Robot connection types** box, find the robot connection type to update.
4. To the right of the robot connection, select **...**, and then select **Edit**.  
The Edit robot connection type panel appears.
5. Update the fields as necessary, and select **Update**.

If the robot connection type is used by a connection, a warning appears. You can cancel or proceed. If your change makes a connection invalid, such as by adding a new field, the status of the connection changes to Draft.

## View HTML and XPath

While building or troubleshooting a robot, you sometimes need to understand the underlying HTML for a page or UI element.

### What Do You Need to Do?

| Goal  | Link  |
|---|---|
| Inspect the HTML for a page                                       | <a href="#">View the HTML Code for a Page</a> |
| Find the XPath value for a UI element                             | <a href="#">View an Element's XPath</a>       |
| Identify all of the XPaths and other elements that you can target | <a href="#">View All Elements to Target</a>   |

## View the HTML Code for a Page

The only way to determine how a UI element is coded is by reviewing its HTML. These instructions are for to the Google Chrome browser.

### Inspect a Page

1. Right-click any element on a web page, and select **Inspect**.

The developer tools appear.

2. Ensure that the **Elements** tab is selected.

This tab shows you the Document Object Model (DOM) of the web page. The DOM represents the web page as nodes and objects and allow the recorder and other tools to interact with the page. You can review the DOM to determine how a UI element is coded.

### Identify the Part of the Page that a Node or Object Specifies

1. Complete the previous steps.
2. Point to any element on the **Elements** tab.

The UI element that the node or object specifies is highlighted in blue. Some nodes and objects apply to the entire page, so they highlight the entire page.

You can also view an element's XPath. See [View an Element's XPath](#).

## View an Element's XPath

The recorder identifies the XML Path Language, or XPath, of the element that a robot takes action on. You can view an element's XPath at any time from within a robot action and from your internet browser.

You typically need to view an element's XPath for the following situations:

- The recorder is unable to target the exact element that you require.  
Get the XPath value yourself, and manually paste the value into the recorder.
- A robot has suddenly started failing, and you need to determine whether the HTML for the page has changed.



You have the following options for viewing an element's XPath:




- [View an XPath from a Browser](#)
- [View an XPath from the Robot Canvas](#)
- [View an XPath within a Robot Action](#)
- [Paste an XPath into a Locator Field](#)

### View an XPath from a Browser

These instructions are for the Google Chrome browser.


1. In a Google Chrome browser, right-click any element on a web page, and select **Inspect**.  
The developer tools appear.
2. Ensure that the **Elements** tab is selected.  
This tab shows you the Document Object Model (DOM) of the web page. The DOM represents the web page as nodes and objects and allow the recorder and other tools to interact with the page. You can review the DOM to determine how a UI element is coded.
3. On the Elements tab of the developer tools, right-click any node or object, point to **Copy**, and select **Copy XPath**.  
To see the value, paste into any text editor.

### View an XPath from the Robot Canvas

1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Along the toolbar, select **Targets** .
3. Select any target.
4. Select **Edit** .
- A panel appears.
5. View the XPath value in the **Locator** field.

### View an XPath within a Robot Action

1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Double-click any action.  
The panel for the robot action appears.

3. Select within the **Locator** field, and select **Show target selector** .

The Target selector field appears below the Locator field. This field shows the XPath value for the target, prefaced by `xpath:.`

### Paste an XPath into a Locator Field

1. Complete any of the previous steps to view an XPath value, and copy the value.
2. Paste the value into a Locator field in a robot action.
3. Preface the value with `xpath:.`

For instance, for the following XPath:

```
//*[@id="buttonName"]
```


Enter the following value:

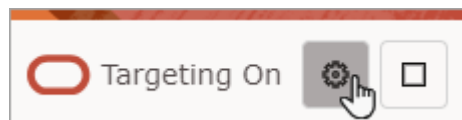
```
xpath://*[@id="buttonName"]
```

## View All Elements to Target

When you add an action to a robot using the low-code tools and use the recorder to select an element in the user interface, you have the option of viewing all elements that you can take action on.

You typically complete this task only if a robot failed during testing, and you want to view all of your options for targeting.

1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.
2. Add an action to a robot using the low-code tools.  
See [Add an Action to a Robot](#).
3. While you use the recorder to define a locator field for the action, select **Settings** in the Smart Recording window in the lower-left corner of your browser.



The Settings panel appears.

4. In the Settings panel, select **Select from various generated selectors**, and close the Settings panel.
5. Point to a UI element that the robot needs to work in, and after the icon changes to a target and the shading turns green, select the UI element.

The Target Element panel appears. This panel lists all the possible XPath values and other elements that you can target for the selected component.

6. Identify the selector to target, and click **Select**.

The recorder enters a value in the **Locator** field in the robot.

The setting remains selected for your sign-in session.

# 5

## Run and Test a Robot

Oracle Integration offers several ways to run and test a robot. Each method offers benefits.

### Options for Testing

| Building stage  | Testing benefit   | Prerequisites  | Testing options   |
|---|---|--|---|
| After you finish building a robot                                   | Replicate real-world conditions by testing a robot in its environment | <a href="#">Specify Where a Robot Runs</a><br><a href="#">Activate a Robot</a> | <a href="#">Test a Robot on Its Environment</a><br><a href="#">Fix a Robot's Errors</a> |
| After you finish building a robot and the integration that calls it | Ensure end-to-end success by testing a robot with its integration     | <a href="#">Design an Integration That Calls a Robot</a>                       | <a href="#">Test a Robot and Its Integration</a>  |

## Workflow for Testing a Robot


Test as you build so that you ensure that the robot completes its task as expected.

Previous workflow: [Workflow for Building a Robot](#).

| Step | Task   | More information  |
|------|--|---|
| 1    | <a href="#">Activate a Robot</a>                 | After you finish building, get your robot ready to run by activating it. You must activate a robot before you can run it in its environment pool. |
| 2    | <a href="#">Test a Robot on Its Environment</a>  | After your robot is finished and you've activated it, you can run the robot.  |
| 3    | <a href="#">Test a Robot and Its Integration</a> | After testing a robot on its own, replicate a real-world scenario by activating the integration and then running the robot from an integration.   |

## Open a Robot

Your first step in reviewing or updating a robot is opening it.

1. In the navigation pane, select **Projects**.
2. Select the project name.
3. In the left toolbar, select **Robot** .
4. In the **Robots** box, select the robot to open.  
The canvas appears.

Next, update the robot as needed. For instance:

- [Add an Action to a Robot](#)
- [Add Logic to a Robot](#)


- [Fix a Robot's Errors](#)

## Fix a Robot's Errors

To fix a robot's errors, you must first validate the robot to identify the errors. Then, determine the cause of each error and fix it.

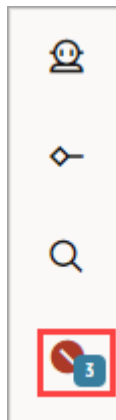
### Validate a Robot

Every time you save a robot, Oracle Integration validates the robot and identifies any errors. You can also validate the robot yourself at any time and then correct any errors that have occurred.

1. Open a robot.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
  - c. In the left toolbar, select **Robot** .
  - d. In the **Robots** box, select the robot to open.  
The canvas appears.

2. On the toolbar, select **Validate model** .

If the validation finds any errors, an error count appears on the right toolbar.



### Fix an Error in a Robot

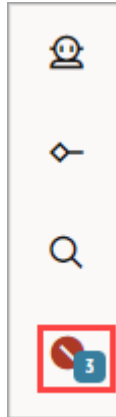
Errors occur in a robot for different reasons, such as an incomplete robot action or a renamed resource. You must fix all errors in a robot before you can activate a robot.

Fixing errors is often an iterative process. Try to fix errors as you build. Keep in mind that the validation process looks for errors in the following way:


1. The validation identifies all **syntax errors** in the robot.
2. After you correct all syntax errors and save the robot again, the validation identifies all **semantic errors** in the robot.

Therefore, you might address all the errors found, save your robot again, and find new errors.

1. Validate a robot.  
See [Validate a Robot](#).
2. On the right toolbar, select **Errors**.



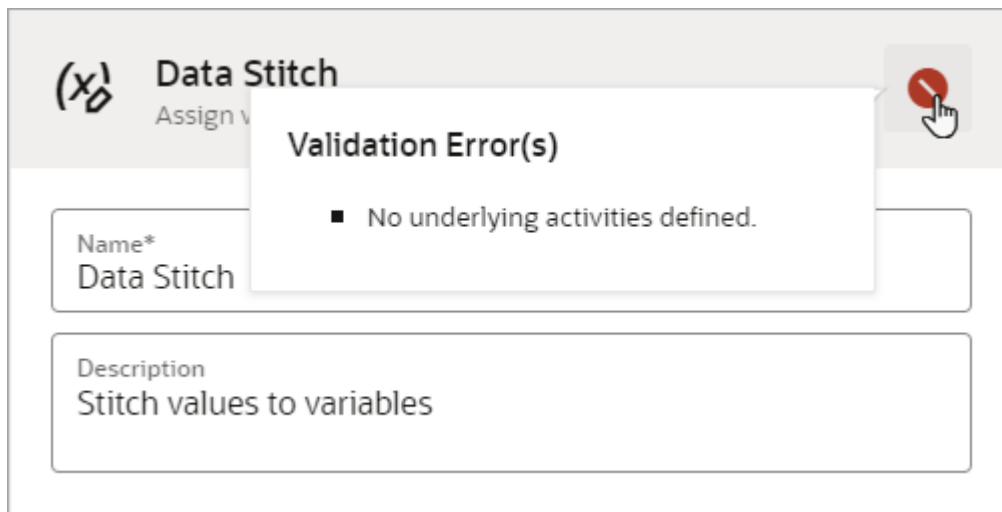
The Errors panel appears.

3. Point to an error, and select **Edit** .

Oracle Integration scrolls to the action on the canvas and opens the panel for the action. The panel and the action contain an **Action has a validation error** icon.



4. To remind yourself of the error, select the **Action has a validation error** icon.  
The Validation Error(s) popup appears.




5. Address the errors as needed.
6. Rerun validation and address any errors until the robot contains no more errors.

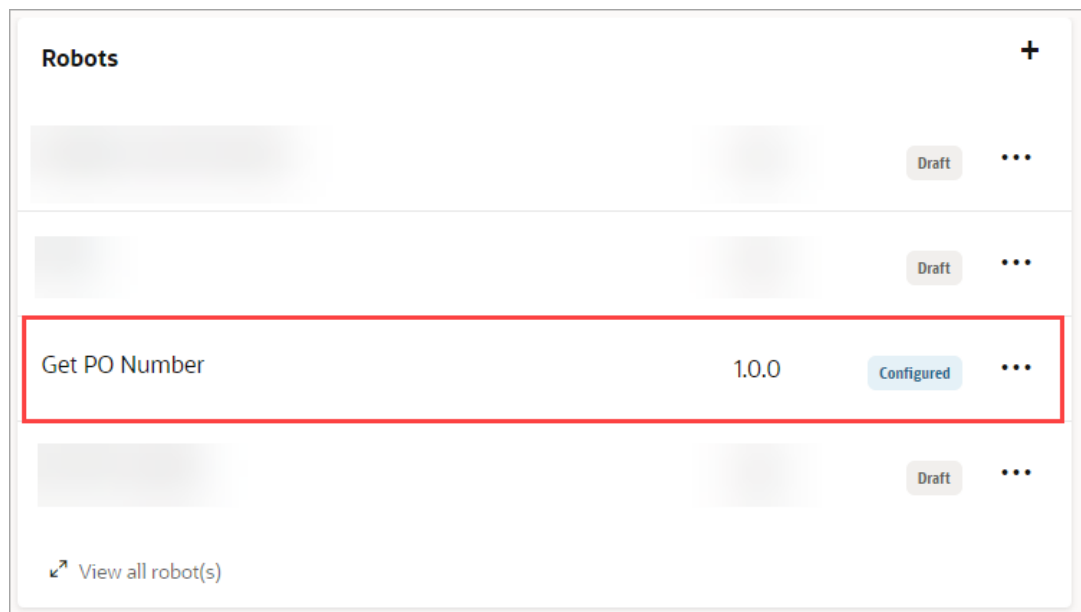
## Activate a Robot

After you finish building a robot, activate it. You must activate a robot to be able to test and run it.

You can have only one major version of a robot active at any given time. For instance, if version 1.0.0 is currently active, you can't make version 1.0.1 or version 1.1 active. However, you can make version 2.0 active.

Before you activate a robot, you must associate the robot with an environment pool. See [Specify Where a Robot Runs](#).

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. In the **Robots** box, ensure that the robot's state is **Configured**.



4. In the **Robots** box, point to the Configured robot, select **...**, and select **Activate**.  
A Confirmation pop-up appears, and the state changes to **Activation in progress**.



5. Navigate to another area of the application, and then return to the project to check the robot's state.

Within a minute or two, the robot's state changes to **Active**.



Next, you can test the robot in its environment. See [Test a Robot on Its Environment](#).

## Test a Robot on Its Environment


After you finish building a robot, test the robot on its real-world environment by running the robot from within the project.

### Prerequisites

- Specify where the robot runs and activate the robot. See [Specify Where a Robot Runs](#) and [Activate a Robot](#).
- If your organization hasn't updated its allowlist to allow outbound calls to Oracle Integration, the robot agent might not be able to contact Oracle Integration so that it can poll for work, and your robot might not run if you're on a company VPN or wired network. See [Review Your Network Configuration](#).
- You don't need to design the integration that calls the robot.

### Test a Robot in Its Environment:

If the robot is associated with an environment pool that contains multiple environments, you can't choose the environment that the robot runs on. If you want to control this aspect of testing, associate the robot with an environment pool that has only one environment.

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. Run the robot.
  - a. In the **Robots** box, point to an Active robot, select **...**, and select **Run**.  
The Configure and run page appears.
  - b. In the **Input** field, enter or paste the JSON code for the trigger's input.

To test your robot effectively, you must enter JSON code that matches the values that you defined in the input.

When the robot runs in the real world, the integration passes this value to the robot. However, you're running the robot independent of its integration, so you must pass this value manually.

### Formatting rules

- Enclose all code in curly brackets: { }
- Include a comma after each line.
- Enclose the property names and the values of strings in quotation marks: " "
- Enclose the values of collection variables in brackets: [ ]
- If a value is a string and is part of a collection variable, include the quotation marks within the brackets. For example:

```
"Property_Name": ["Value"],
```

### Sample code: Input with multiple properties



The input for a robot contains a number of properties, including `Department`, `Name`, `Activity_Type`, and so on, as shown in the following example. You can pass values for these properties to the robot.

```
{
  "Department": "Finance",
  "Name": "Travel expenses",
  "Activity_Type": ["Adjustments"],
  "Active": "Yes",
  "Account_Source": ["Travel account"],
  "Tax_Rate_Code_Source": ["Activity"],
  "Activity_Account": "ABCD-123456-789000",
  "searchTask": "manage expenses"
}
```

### Sample code: Passing multiple records for an input that is based on a data type with four properties

The input for a robot is named `Invoices` and is of the `InvoiceType` type. The `InvoiceType` data type contains the following properties:

- `InvoiceNumber`
- `SupplierName`
- `InvoiceAmount`
- `DueAmount`

The input must include the name of the input, the name of its properties, and the values for its properties. If the robot iterates over multiple records, you can pass multiple records to it in the input, as shown in the following example.

```
{
  "Invoices": [{
    "InvoiceNumber": "US123456",
    "SupplierName": "Big Computers",
    "InvoiceAmount": 98437,
    "DueAmount": 500.00
  }, {
    "InvoiceNumber": "US234566",
    "SupplierName": "Small, Inc.",
    "InvoiceAmount": 128,
    "DueAmount": 128
  }]
}
```

**c. Select `Run`.**

Below Instance ID, the identifier of the robot instance appears.

Next, monitor the robot to verify that it ran as expected.

After verifying that the robot runs successfully in its environment, test the robot with its integration. See [Test a Robot and Its Integration](#).

## Test a Robot and Its Integration

After ensuring that the robot runs as expected, test the robot along with the integration that calls it. This end-to-end testing also helps you confirm that the integration passes the correct values to the robot.

### Prerequisites

[Activate the robot.](#)


If an integration tries to call a robot that isn't active, the integration fails.

### Workflow

1. [Activate an Integration](#)
2. [Run an Integration and Robot](#)

## Activate an Integration

After you finish designing the integration that calls your robot, you must activate it and set its tracing level.

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the **Integrations** section, find the integration to activate.
3. Click **Actions** , and select **Activate**.


The Activate integration panel opens.
4. Follow the steps to activate the integration.

See [Activate and Deactivate Integrations](#) in *Using Integrations in Oracle Integration 3*.

## Run an Integration and Robot

After you activate a robot and the integration that calls it, you can run them both by running the integration.

1. Open a project.
  - a. In the navigation pane, select **Projects**.
  - b. Select the project name.
2. In the **Integrations** box, find the integration to activate.

The integration must have an Active label.
3. Click **Actions** , and select **Run**.

The Configure and run page opens.
4. Click **Run**.

The Activity Stream panel opens to show the progress of the run.

# 6

## Troubleshoot

Get help troubleshooting the robot agent, the recorder, and robots.

### What Do You Need Help With?

- [Troubleshoot the Robot Agent and Environments](#)
- [Troubleshoot the Recorder](#)
- [Troubleshoot Robots](#)
- [Download the Log File for a Robot or Robot Agent](#)

## Troubleshoot the Robot Agent and Environments

Get help when you can't install or start the robot agent, or when an environment is unavailable.

### What Are You Trying to Do?

| Goal                             | Troubleshooting help                       |
|----------------------------------|--|
| Start or restart the robot agent | <a href="#">Cannot Start a Robot Agent</a> |

## Cannot Start a Robot Agent

If you're unable to start running a robot agent, whether for the first time or after stopping it, use the troubleshooting steps to get back on track.

### Issue

You cannot stop the robot agent, either immediately after you installed it or after someone stopped it.

### Why It Happens

This issue typically occurs when the robot agent can't connect to Oracle Integration, or when the configuration of the robot agent requires some adjustments. Keep reading for step-by-step instructions on how to troubleshoot.

### What to Do

After completing each step, try running the agent again.

| Step | Consideration   | More information  |
|------|---|---|
| 1    | Did an error occur when you tried to start the agent?                             | <p><b>Error text</b></p> <p>User agent set to: Oracle-JavaSDK/2.56.0<br/>           (Mac OS X/14.3.1; Java/17.0.8; Java HotSpot (TM) 64-Bit Server VM/17.0.8+9-LTS-211)<br/>           Exception in thread "main"<br/>           com.oracle.bmc.model.BmcException: Error returned by GetAgentVersion operation in Agent service.(401, NotAuthenticated, false)<br/>           The required information to complete authentication was not provided or was incorrect. (opc-request-id: <i>debugging_identifier/debugging_identifier/debugging_identifier</i>)<br/>           Timestamp: <i>timestamp</i><br/>           Client version: <i>robot_agent_version</i><br/>           Request Endpoint: <i>endpoint</i><br/>           Troubleshooting Tips: See <a href="https://docs.oracle.com/en-us/iaas/Content/API/References/apierrors.htm#apierrors_401_401_notauthenticated">https://docs.oracle.com/en-us/iaas/Content/API/References/apierrors.htm#apierrors_401_401_notauthenticated</a> for more information about resolving this error<br/>           Also see for details on this operation's requirements.<br/>           To get more info on the failing request, you can enable debug level logs as mentioned in `Using SLF4J for Logging section` in <a href="https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/javasdkconfig.htm">https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/javasdkconfig.htm</a>.<br/>           If you are unable to resolve this Agent issue, please contact Oracle support and provide them this full error message.</p> <pre>           at           com.oracle.bmc.http.internal.ResponseHelper.throwIfNotSuccessful(ResponseHelper.java:160)           </pre> <p><b>What to do</b></p> <p>Navigate to the folder that contains the unzipped files for the robot agent, and delete the <code>Agent.properties</code> file.<br/>           Next, try starting the agent again.</p> |
| 2    | Does the robot agent's computer meet the system requirements for the robot agent? | <p>For instance, the correct version of the JDK might not be installed, or the <code>JAVA_HOME</code> might not be set.<br/>           See <a href="#">System Requirements</a>.</p>   |
| 3    | Does the robot agent's computer have internet access?                             | <p>Navigate to a website that your network allows you to access, and confirm that the page loads.<br/>           If the computer doesn't have internet access, work with a network administrator to correct this issue.</p>   |

| Step | Consideration  | More information   |
|------|--|--|
| 4    | Does your network allow the robot agent's computer to access Oracle Integration?     | Sign in to Oracle Integration from the computer that the robot agent is installed on.<br>If you are unable to sign in, your network might restrict access to the internet based on IP address.<br>Work with an administrator to ensure that the egress rules allow access to the inbound IP address for Oracle Integration. Otherwise, the robot agent cannot access Oracle Integration.<br>See <a href="#">Obtain the Inbound and Outbound IP Addresses of the Oracle Integration Instance in Provisioning and Administering Oracle Integration 3</a> . |
| 5    | Does a confidential application exist for the robot agent, and is it active?         | A confidential application is an OAuth client application that allows robot agents to securely connect to Oracle Integration using the OAuth protocol.<br>See <a href="#">Ensure that the Confidential Application is Active</a> .   |
| 6    | Is the robot agent's configuration file defined correctly?                           | For step-by-step instructions on updating the file, see <a href="#">Update the Robot Agent's Configuration File</a> .  |
| 7    | Is there a space in the folder name or directory where the robot agent is installed? | Do not include any spaces in the folder name, or you cannot start the robot agent.<br>Additionally, Oracle recommends not including any spaces in the directory path.  |
| 8    | Enter a service request  | If you complete all of these steps and the robot agent still won't run, get help from Support by entering a service request (SR). Include the log files for the robot agent with the request. See <a href="#">Download the Log File for a Robot or Robot Agent</a> .   |

## Troubleshoot the Recorder

Get help when the screen recorder that you use to build a robot doesn't work the way that you expect.

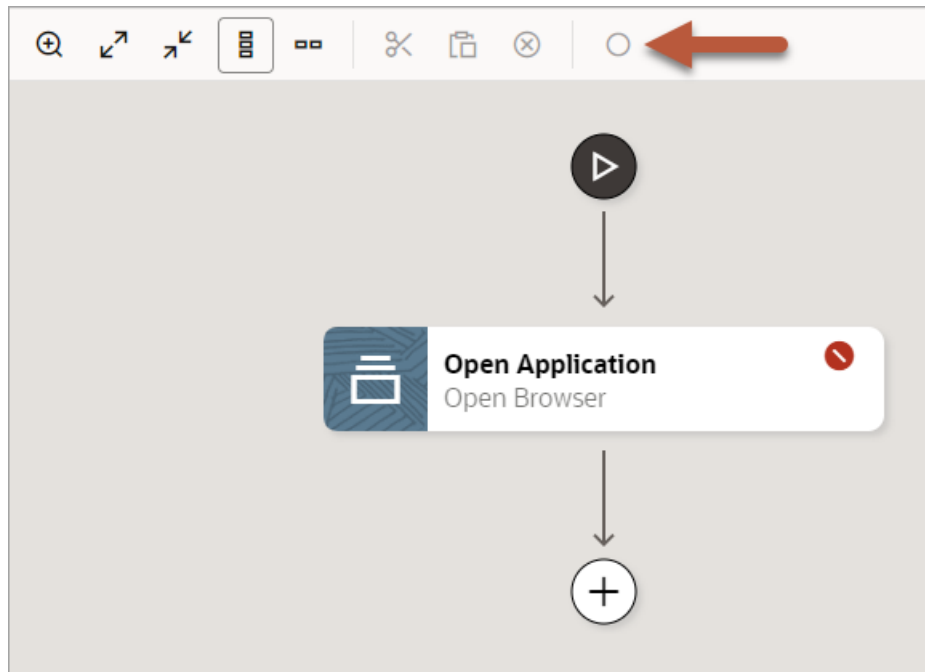
| Goal                             | Troubleshooting help   |
|----------------------------------|--|
| Start the recorder               | <a href="#">Can't Start the Recorder</a>   |
| Build a robot using the recorder | <a href="#">Recorder Suddenly Stops Working</a><br><a href="#">Application Isn't in Browsers List</a><br><a href="#">Canvas Is Read-Only</a> |

## Can't Start the Recorder

If you can't start the recorder from the canvas, make sure that you've selected an action on the canvas.

### Issue

The **Record after the selected action** button  is grayed out on the canvas, so you can't start recording.

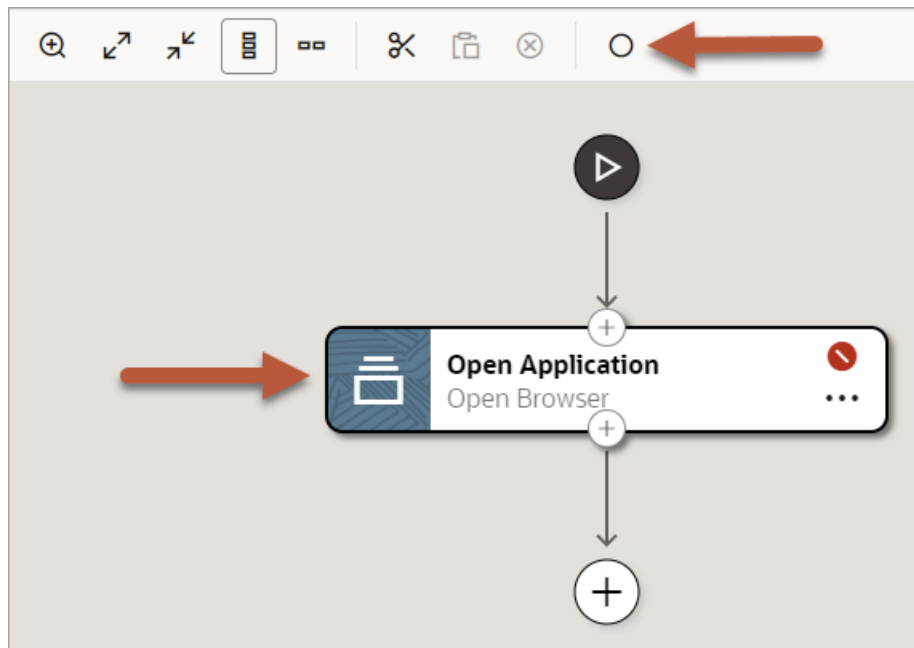


### Why It Happens

The button becomes enabled only when you select an action in the canvas.

### What to Do

Select an action in the canvas.



## Recorder Suddenly Stops Working

When you're building a robot using the recorder, and the recorder suddenly stops working, a significant change to the page's HTML is often the reason why.

### Issue



You're in the middle of recording, but you suddenly can't target any UI elements. The targeting colors no longer appear when you point to various UI elements.


### Why It Happens

This issue occurs when a page replaces its HTML with new HTML without loading a new page. The recorder is unable to continue recording because the page hasn't changed--but its HTML has.

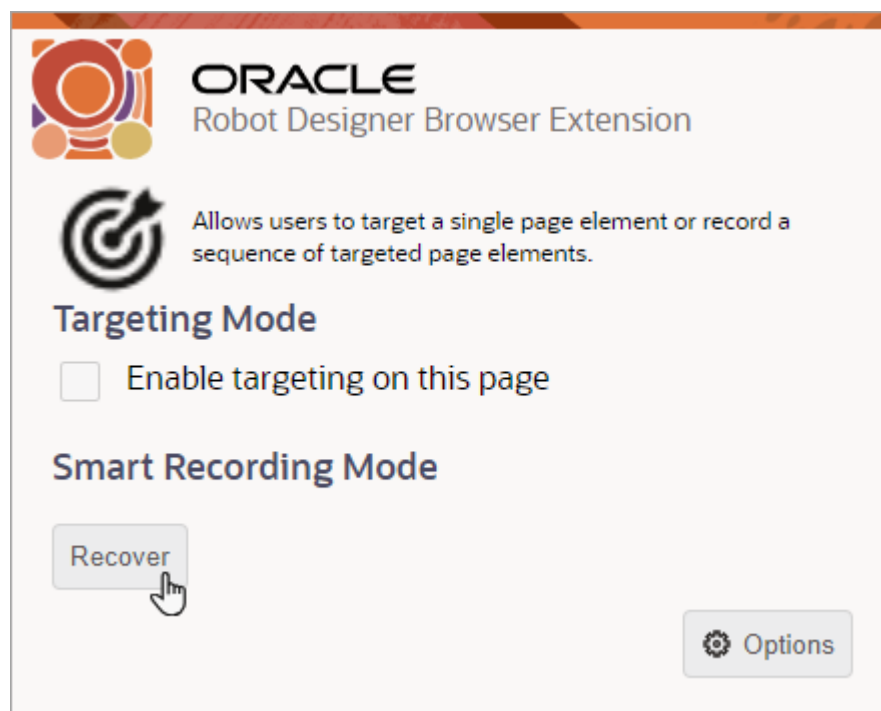
### What to Do

You have several options for moving forward:

- Return to Oracle Integration, stop the recording by selecting **Stop**  on the toolbar, and restart the recording from the same position in the robot where you left off.  
Next, continue with your recording.
- In the toolbar of the browser that you're recording in, select the icon for the Robot Designer Browser Extension .

This icon appears with your other extensions. For example, in Chrome, you might need to select Extensions  to see it.

Next, select **Recover**.



Next, continue with your recording.

## Application Isn't in Browsers List

When an application doesn't appear in the list of browsers to target, the application might not have been open when you started the recorder.

### Issue

When you're using the targeting tool to identify a locator field for a robot action, your application's browser is sometimes missing from the list.

### Why It Happens

This issue typically occurs when you open the application after you start the recorder.

### What to Do

Stop the recorder, open the application in a new tab, and start the recorder again.

#### Tip:

If you opened the application before starting the recorder, make sure that you're scrolling down all the way in the list of browsers. If you have a number of browsers open, you might need to scroll all the way to the bottom of the list.

## Canvas Is Read-Only

If everything is read-only, you're probably still recording.

### Issue

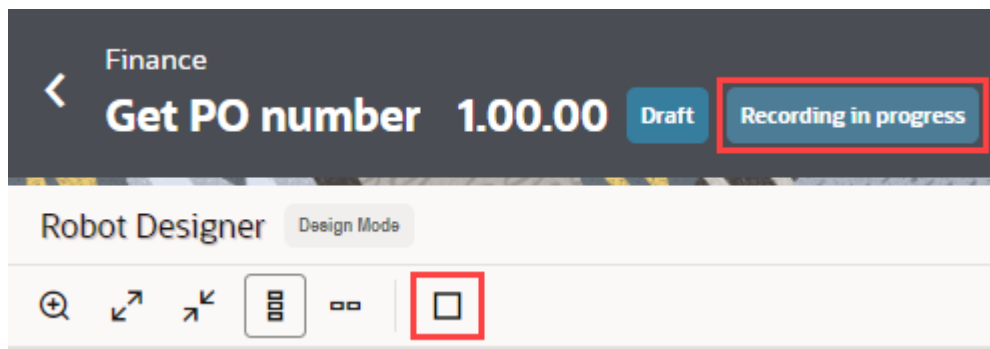
Most actions on the canvas are read-only or unavailable.

### Why It Happens

Most actions are read-only or unavailable while the recorder is running.

### What to Do

If you can't complete any tasks, check whether the **Recording in progress** badge and the **Stop recording** buttons appear. If so, you're still recording.



Stop the recording, and then you can work on the canvas.



# Troubleshoot Robots

Get help troubleshooting robots.

## What Are You Trying to Do?

| Goal             | Troubleshooting help   |
|------------------|--|
| Activate a robot | <a href="#">Can't Activate a Robot</a>   |
| Run a robot      | <a href="#">Robot Failed</a><br><a href="#">Robot Action Takes a Long Time</a> |

## Can't Activate a Robot

If you can't activate a robot, the robot might still contain an error, or you might not have associated an environment with the robot.

### Issue

You are unable to activate a robot.

### Why It Happens

This issue typically occurs when:

- The robot contains one or more errors.
- The robot isn't associated with an environment.

### What To Do

- Determine whether any errors are present.  
See [Fix an Error in a Robot](#).
- Determine whether the robot is associated with an environment.  
See [Create an Environment Pool](#).

## Robot Failed

When a robot fails, use the troubleshooting steps to get back on track.

### Issue

A robot failed.

## Why It Happens, and What To Do

| Step | Why it happens   | More information   | What to do   |
|------|--|--|--|
| 1    | The robot doesn't have the required access   | <p>You typically build a robot using different credentials than a robot uses to sign in to an application. Your credentials might have more or different permissions than the robot.</p> <p>For instance, when you built the robot, your credentials might have shown you pages and buttons that the robot can't see due to its more limited access.</p>   | Verify that the robot's credentials have the required access to complete its tasks.  |
| 2    | The environment pool that the robot is associated with doesn't have any environments, or none of its environments are active |  | If the environment pool has no environments, add one or more environments to it. See <a href="#">Specify Where a Robot Runs</a>  |
| 3    | The robot timed out  | <p>A robot can time out in the following ways:</p> <ul style="list-style-type: none"> <li>The total running time for the robot exceeds the service limit. This timeout typically occurs due to a programming error. For instance, the robot might create an infinite loop. To address this issue, determine the action where the robot got stuck, and make the necessary corrections.</li> <li>A prevalidation page state on an action within the robot times out. A prevalidation page state specifies a timeout period. If the timeout period passes before the page state's requirements are met, the action doesn't occur, and the robot times out. See <a href="#">Add Validation to a Robot Action</a>.</li> <li>A "wait until element is visible" action within the robot times out. A wait action specifies a timeout period. If the timeout period passes before the action's requirements are met, the robot times out. See <a href="#">Add a Wait Until Element Is Visible Action</a>.</li> </ul> | <p>You might or might not need to address timeouts. For instance, if a robot has been running successfully for months and has failed because the application that it works in has gone down or because the network experienced temporary latency, the robot runs successfully again after the issues are resolved.</p> <p>However, if a robot consistently times out during testing, latency issues are often to blame. Complete the robot's task yourself in the application, and then consider how to address the issue. For instance, increasing the timeout period might prevent timeouts.</p> |

| Step | Why it happens   | More information  | What to do  |
|------|--|---|---|
| 4    | The page's underlying HTML has changed since you built the robot | If a robot has run successfully for a period of time and now consistently fails on the same step, the application's underlying HTML for the page might have changed.  | Review the XPath of the element in the action that has failed, and compare it to the XPath in the robot. See <a href="#">View an Element's XPath</a> .  |
| 5    | The HTML code for an element doesn't match its action            | If you used the low-code tools to build your robot, you might have chosen an action that can't interact with the underlying HTML code. For instance, sometimes a UI field looks like a drop-down list but is actually coded as a text field.<br><br>For example, the following actions are used for similar activities: <ul style="list-style-type: none"> <li>• <a href="#">Add a Checkbox Action</a></li> <li>• <a href="#">Add a Click Element</a></li> <li>• <a href="#">Add an Enter Text Action</a></li> <li>• <a href="#">Add a List Action</a></li> </ul> | Use the recorder to recapture the action that is failing. When you use the recorder to create robot actions, the recorder reviews the page's HTML and determines the appropriate robot action to use.   |
| 6    | The action targets the wrong UI element                          | While identifying the field that a robot needs to act on, you might have inadvertently targeted the wrong element.  | The advanced settings for the recorder allow you to view all the elements to target before saving a specific XPath. See <a href="#">View All Elements to Target</a> .<br><br>Alternatively, if the recorder is unable to target the exact element that you require, you can obtain the XPath of the element, and paste the value into the recorder. See <a href="#">View an Element's XPath</a> . |

## Robot Action Takes a Long Time

When a robot action always takes longer than expected, a fixed time validation might be to blame.

### Issue


A robot runs slower than expected, and the same action always takes longer than expected.

### Why It Happens

The action might have validation that includes a mandatory wait time.

### What To Do

#### Step 1. Identify the action that is taking a long time

1. Open the project that contains the slow-running robot.
2. Select the **Observe** tab.
3. Select the **Robot instances** tab.
4. In the list of robot instances, find a recent instance of the robot.
5. Point to the robot instance, and select **View details** .

The Activity stream opens for the robot instance.



6. Review the timestamp for each action, and determine the action that is taking longer than expected.



 **Note:**

If the robot started later than expected, the environments that it runs on might have been busy running other robots. Ensure that you have dedicated enough environments for your robots. For guidance on processing bulk data, see [Processing Bulk Data Best Practices](#).

### Step 2. Determine whether the action has any validation

1. Open the project that contains the slow-running robot.
2. Open the robot.
3. Double-click the action that you identified from the activity stream.
4. In the panel for the action, select the **Pre Validate** tab.
5. If a page state is selected in **Page state** drop-down, review its validation.
  - a. Above the **Page state** drop-down list, select **Edit** .
  - b. From the **Conditions** list, select a validation condition to review, and select **Edit** .
  - c. Review the selection for the **Strategy** drop-down list.
    - If **Fixed Time** is selected, the validation includes a mandatory wait time. The mandatory wait time is likely the cause of the slow-running robot.
    - If **Element** is selected, the validation waits up to a specific amount of time for an element to be visible or enabled.

If the application that the robot works in experiences performance issues, this validation is doing its job: allowing the robot to wait until a UI element is visible before it proceeds.

## Download the Log File for a Robot or Robot Agent

If you need help troubleshooting issues with a robot or robot agent, enter a service request (SR) and include the log file. Downloading the file takes just a couple moments.

1. Navigate to the appropriate location:
  - For a robot instance:

```
<UserHome>/ .orpa/instances/<InstanceId>/output/
```
  - For the robot agent:

*Robot\_agent\_installation\_path/Robot\_agent\_folder/orpa-agent-  
service-logs/info-Date.log*

2. Enter an SR, and attach the log file to the request.

# A

## Use Cases

Explore how to implement real-world scenarios.

### What Do You Want to Learn About?

- [Use Case: Update a Set of Invoices](#)
- [Use Case: Save Data After Iterating](#)
- [Use Case: Switch Browsers](#)

## Use Case: Update a Set of Invoices

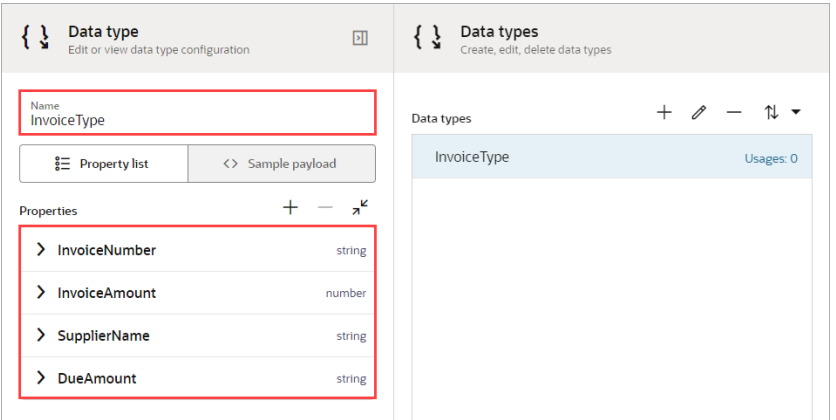
A robot must manually update a set of invoices. Explore how data types, triggers, a foreach loop, and expression syntax support this workflow.

### Scenario

A robot completes the following tasks for a set of invoices:

- Identifies an invoice to update in Oracle Cloud ERP.
- Finds the invoice.
- Verifies the supplier name and invoice totals, and then updates the invoice.

### Workflow

| Requirement                           | How to meet the requirement   |
|---------------------------------------|---|
| Define the components of invoice data | <p>Create a data type, <code>InvoiceType</code>, with the following properties:</p> <ul style="list-style-type: none"><li>• <code>InvoiceNumber</code></li><li>• <code>InvoiceAmount</code></li><li>• <code>SupplierName</code></li><li>• <code>DueAmount</code></li></ul>  <p><b>Tip:</b> To learn more about data types and other placeholder values, see <a href="#">Alternatives to Hard Coding Data</a>.</p> |

| Requirement        | How to meet the requirement   |
|--------------------|---|
| Define the trigger | Create an input, InvoiceCollection, that is of the InvoiceType data type.<br>Make the trigger a collection so that it can pass in an array of data--in this case, a number of invoices. |

**Trigger**  
Define the robot input & output signature

**Input**      **Output**

Properties + - ↗

**InvoiceCollection**      InvoiceType collection

Name  
InvoiceCollection

Type  
InvoiceType

Collection

|  |  |
|--|--|
| Define places to store invoice amount and supplier name data | Create the following variables: <ul style="list-style-type: none"> <li>• InvAmt</li> <li>• SuppName</li> </ul> |
|--|--|

**(x) Variables**  
Create, edit, delete variables

Variables + ✎ - ↕ ▼

|                          |          |        |           |
|--------------------------|----------|--------|-----------|
| <input type="checkbox"/> | InvAmt   | string | Usages: 0 |
| <input type="checkbox"/> | SuppName | string | Usages: 0 |

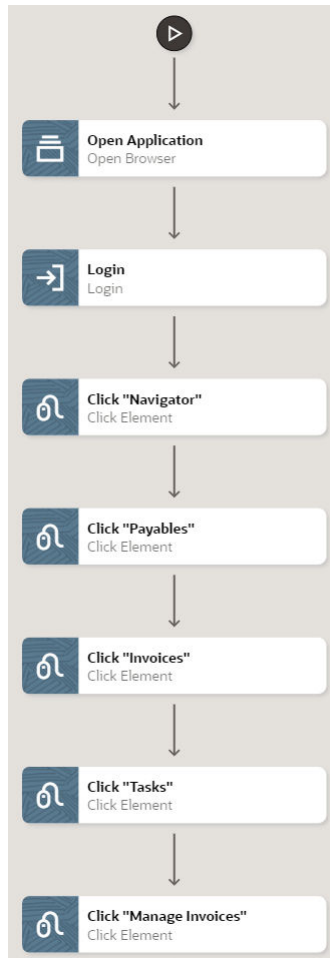
---

| Requirement | How to meet the requirement |
|-------------|-----------------------------|
|-------------|-----------------------------|

---

Define the navigation to the page in the application where you search for invoices

Using either the recorder or the low-code tools, define the actions that allow the robot to navigate to the area of the application for finding invoices. After you finish defining the actions, the robot looks something like this:





| Requirement | How to meet the requirement |
|-------------|-----------------------------|
|-------------|-----------------------------|

Create a container for the actions that the robot must perform on every invoice

Add a foreach loop to the robot. The foreach loop defines how the robot iterates over the invoices.  
In the foreach loop, use the **Collection** field to identify the data set that the foreach loop iterates on. Insert the input, `InvoiceCollection`, into this field.  
The **Iteration parameter**, `CurrentInvoice`, is the name to use for every record in the collection. This entry becomes a variable that you can reference only in actions within the foreach loop.

**foreach**  
Execute actions for each element of a data set

Name\*  
foreachInvoice

Description

Collection\*  
\$\${INPUT.InvoiceCollection}

Iteration parameter\*  
CurrentInvoice

Record the invoice number in the activity stream

Within the foreach loop, create a log action to create an entry in the activity stream for logging the invoice number of every invoice that the robot updates. Knowing the invoice number is helpful if you need to troubleshoot a robot instance.

**(x) Variables**  
Drag and drop a variable or a property of the variable

Available variables +

- InvAmt string
- SuppName string
- CurrentInvoice InvoiceType
  - InvoiceNumber string
  - InvoiceAmount number
  - SupplierName string
  - DueAmount string

**Log**  
Add a message to the robot flow activity stream

Name\*  
Log

Description  
Add a message to the robot flow activity stream

Message\*  
\$\${VARIABLE.CurrentInvoice[InvoiceNumber]}

**Requirement**

**How to meet the requirement**

Enter the invoice number into the Search field in the application

Within the foreach loop, use the recorder or low-code tools to enter the value from the `InvoiceNumber` property of the `CurrentInvoice` variable into the Search field in Oracle Cloud ERP.

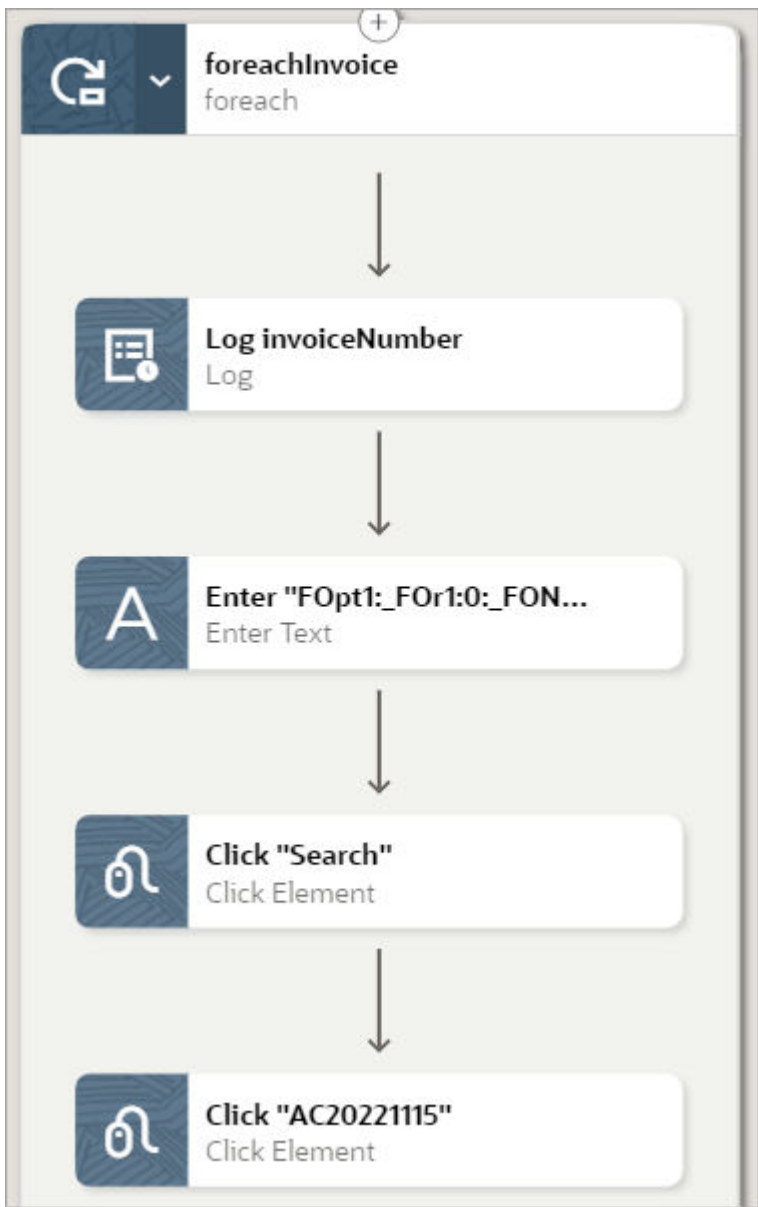
The screenshot displays the Oracle Integration Studio interface. On the left, the 'Variables' panel shows a tree view with 'CurrentInvoice' expanded, and 'InvoiceNumber' selected and highlighted with a red box. The 'Action Details' panel on the right shows an 'Enter Text' action. The 'Value' field is populated with the expression `${VARIABLE.CurrentInvoice[InvoiceNumber]}`, which is also highlighted with a red box. The 'Target name' is set to 'Order' and the 'Element' is 'INPUT'. The 'Test value' field contains 'Test data'. At the bottom right, there are 'Save' and 'Discard' buttons.

**Requirement**

**How to meet the requirement**

Open the invoice

Within the foreach loop, define additional required interactions to open an invoice.



**Requirement**

**How to meet the requirement**

Get the supplier name from an invoice, and save it to a variable

Within the foreach loop, use the recorder or low-code tools to get the supplier name from an invoice and save the value to the `SuppName` variable.

The screenshot shows the Oracle Integration Studio interface. On the left, the 'Variables' panel lists 'InvAmt string', 'SuppName string' (highlighted with a red box), and 'CurrentInvoice InvoiceType'. On the right, the 'Action Details' panel shows 'Name' as 'Get Text', 'Target name' as 'xpath\_\_\_\_id\_pt1\_FOR1\_1\_FONsr2\_0\_MAT2\_0\_', 'Element' as 'TD', and 'Action' as 'Get Text'. The 'Save to' field contains the expression `${VARIABLE.SuppName}` (highlighted with a red box). Buttons for 'Save' and 'Discard' are at the bottom right.

Get the invoice total from an invoice, and save it to a variable

Within the foreach loop, use the recorder or low-code tools to collect the invoice total from an invoice.

Save the value to the `InvAmt` variable.

The screenshot shows the Oracle Integration Studio interface. On the left, the 'Variables' panel lists 'InvAmt string' (highlighted with a red box), 'SuppName string', and 'CurrentInvoice InvoiceType' which is expanded to show 'InvoiceNumber string', 'InvoiceAmount number', 'SupplierName string', and 'DueAmount string'. On the right, the 'Action Details' panel shows 'Name' as 'Get Text', 'Target name' as 'xpath\_\_\_\_id\_pt1\_FOR1\_1\_FONsr2\_0\_MAT2\_0\_', 'Element' as 'TD', and 'Action' as 'Get Text'. The 'Save to' field contains the expression `${VARIABLE.InvAmt}` (highlighted with a red box). Buttons for 'Save' and 'Discard' are at the bottom right.

**Requirement**

Take the appropriate action, depending on whether the company name on the invoice matches the company name in Oracle Cloud ERP

**How to meet the requirement**

Within the foreach loop, create a switch condition that performs the following tasks:

- Condition 1 determines whether the company names match, and if they do, updates the invoice amount.

The first condition contains the following expression in the Condition field:

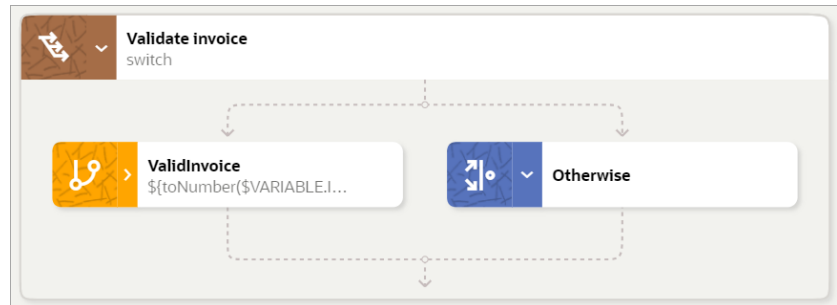
```

${toString($VARIABLE.CurrentInvoice[InvoiceAmount])
== $VARIABLE.InvAmt
&& $VARIABLE.CurrentInvoice[SupplierName]
== $VARIABLE.SuppName}
    
```

In plain English, this condition says the following:

- Return a string value for the invoice amount that was passed into the robot, and compare the value to the invoice amount that the robot obtained from the invoice.
- Compare the supplier name that was passed into the robot to the supplier name that the robot obtained from the invoice.
- If the invoice amounts match AND the supplier names match, update the invoice amount appropriately.

- Condition 2 specifies that if the company names don't match, do nothing.



## Use Case: Save Data After Iterating

A robot must manually cancel a set of invoices and obtain the invoice number and status for each invoice. Explore how a foreach loop and the data stitch action support this workflow.

### Scenario

You must periodically cancel a number of invoices. An integration generates a report that finds the invoices that need canceling. The report provides the invoice numbers of the invoices that need canceling in an XML response.

The robot completes the following tasks:

- Cancels a set of invoices one by one.
- Obtains the invoice number and status for each canceled invoice.

This use case focuses on how you use the data stitch action to obtain the invoice number and status from each invoice. For information about how to update a set of invoices, see [Use Case: Update a Set of Invoices](#).

### Why Create a Robot?

An integration provides improved scalability over a robot for work like this, but you cannot design an integration to cancel the invoices. Here's why: The REST APIs for the application don't allow you to select only invoices of a specific type, and you must cancel invoices with a specific invoice type. Therefore, a robot is the ideal solution for automating this manual, repetitive task.

### Workflow

| Requirement                           | How to meet the requirement  |
|---------------------------------------|--|
| Create the data types for the trigger | <p><b>Define the data type for the input:</b></p> <ul style="list-style-type: none"> <li>• Name: InvoiceNumber</li> <li>• Property: Invoice, of type string, not a collection</li> </ul> <p><b>Define the data type for the output:</b></p> <ul style="list-style-type: none"> <li>• Name: Result</li> <li>• Properties: <ul style="list-style-type: none"> <li>– InvoiceNumber, of type string, not a collection</li> <li>– Status, of type string, not a collection</li> </ul> </li> </ul> |

| Requirement          | How to meet the requirement   |
|----------------------|---|
| Define the trigger   | <p><b>Define the trigger's input:</b></p> <ul style="list-style-type: none"> <li>• Name: InvoiceNumber</li> <li>• Type: InvoiceNumber</li> <li>• Collection: Yes</li> </ul> <p>The input allows the robot to receive a list of invoices from the integration that calls the robot. For example, the robot might receive the invoice numbers for 20 invoices.</p> <p><b>Define the trigger's output:</b></p> <ul style="list-style-type: none"> <li>• Name: Status</li> <li>• Type: Result</li> <li>• Collection: Yes</li> </ul> <p>The output allows the robot to collect the following information so that it can pass the information back to the integration:</p> <ul style="list-style-type: none"> <li>• The invoice number of each canceled invoice.</li> <li>• The status of each canceled invoice.</li> </ul> <p>The output collects information for multiple invoices, so it must be a collection.</p> |
| Define the variables | <p><b>Define a variable to hold the number of each updated invoice:</b></p> <ul style="list-style-type: none"> <li>• Name: CurrInvoice</li> <li>• Type: InvoiceNumber</li> <li>• Collection: No</li> </ul> <p><b>Define a variable to hold the status of each updated invoice:</b></p> <ul style="list-style-type: none"> <li>• Name: CurrentInvoiceResult</li> <li>• Type: Result</li> <li>• Collection: No</li> </ul>   |

**Requirement**

**How to meet the requirement**

Within the robot, add a foreach loop and a data stitch action within it

When a robot must perform the same work on multiple items, define the robots actions in a foreach loop.

This use case is focused on the tasks that you perform in the data stitch and doesn't provide details about all of the actions in the foreach loop. For a use case that focuses on how to update a set of invoices, see [Use Case: Update a Set of Invoices](#).

The foreach loop might look something like this:





**Requirement**

**How to meet the requirement**

Define the details of the data stitch

The following foreach loop contains the following operations:

The screenshot shows the 'Data Stitch' configuration window with the following details:

- Name:** Data Stitch
- Description:** Stitch values to variables
- Operation 1:** Assign. Variable: `$$VARIABLE.CurrentInvoiceResult[InvoiceNumber]`. Value: `$$VARIABLE.CurrInvoice[Invoice]`.
- Operation 2:** Assign. Variable: `$$VARIABLE.CurrentInvoiceResult[Status]`. Value: "Success".
- Operation 3:** Append. Collection: `$$OUTPUT.Status`. Value: `$$VARIABLE.CurrentInvoiceResult`.

- **First operation: Assign a value**

`$$VARIABLE.CurrentInvoiceResult[InvoiceNumber]`  
= `$$VARIABLE.CurrInvoice[Invoice]`

The foreach loop cancels each invoice, one at a time. This assignment loads the invoice number for each canceled invoice to the `InvoiceNumber` object in the `CurrentInvoiceResult` variable.

Because each iteration of the foreach loop loads a new invoice number, the `InvoiceNumber` object in the `CurrentInvoiceResult` variable holds each invoice number only temporarily. The third operation in the data stitch records the value permanently.

- **Second operation: Assign a value**

`$$VARIABLE.CurrentInvoiceResult[Status]`  
= "Success"

This assignment assigns the `Success` value for each canceled invoice to the `Status` object in the `CurrentInvoiceResult` variable.

Because each iteration of the foreach loop loads a new status, the `Status` object in the `CurrentInvoiceResult` variable holds each

| Requirement | How to meet the requirement   |
|-------------|---|
|             | <p>status only temporarily. Another operation in the data stitch records the value permanently.</p> <ul style="list-style-type: none"> <li>• <b>Third operation: Append a value</b><br/> <math>\\${\\$OUTPUT.Status} +</math><br/> <math>\\${\\$VARIABLE.CurrentInvoiceResult}</math></li> </ul> <p>This assignment appends the two objects in the <code>CurrentInvoiceResult</code> property, <code>InvoiceNumber</code> and <code>Status</code>, to the <code>Status</code> output. Because this operation is an append, the operation records all values for all invoices to the output.</p> |

## Use Case: Switch Browsers

A robot must resume working in an already-open internet browser after starting to work in a second internet browser. Explore how the open browser and switch browser actions support this workflow.

### Scenario

A robot completes the following tasks:

1. Opens an internet browser.
2. Completes a task.
3. Opens another internet browser.
4. Completes a task.
5. Resume working in the first internet browser

## Workflow

---

**Requirement**

Add two open browser actions to the robot, as well as any additional actions

**How to meet the requirement**

Build the robot as required, as shown in the following example.



| Requirement | How to meet the requirement |
|-------------|-----------------------------|
|-------------|-----------------------------|

|  |  |
|--|--|
| Use a variable to define the output for each open browser action | <p>For example, create two variables, <code>index1</code> and <code>index2</code>, to store the index value for each open browser action.</p> <p>The index is an assigned number. The index for the first open browser action in the robot is 1, the index for the second browser action in the robot is 2, and so on.</p> |
|--|--|

| (x) Variables   | Open Browser  |
|---|---|
| <p>Drag and drop a variable or a property of the variable</p> <p>Available variables</p> <ul style="list-style-type: none"> <li>index1 string</li> <li>index2 string</li> </ul> | <p>Opens a browser session</p> <p>Name*<br/>Open Application</p> <p>Description<br/>Open browser page.</p> <p>Input Output Post Validate</p> <p>Save to<br/>\${VARIABLE.index1} (x) ...</p> |

| (x) Variables   | Open Browser   |
|---|--|
| <p>Drag and drop a variable or a property of the variable</p> <p>Available variables</p> <ul style="list-style-type: none"> <li>index1 string</li> <li>index2 string</li> </ul> | <p>Opens a browser session</p> <p>Name*<br/>Open Browser</p> <p>Description<br/>Opens a browser session</p> <p>Input Output Post Validate</p> <p>Save to<br/>\${VARIABLE.index2} (x) ...</p> |

|   |   |
|---|---|
| When the robot needs to switch back to the first internet browser, add a switch browser action to the robot | Define the <b>Browser window</b> field using the <code>index1</code> variable. Alternatively, you can hard-code a value of 1, as long as you're switching back to the first browser window that you opened. |
|---|---|

| (x) Variables   | Switch Browser   |
|---|--|
| <p>Drag and drop a variable or a property of the variable</p> <p>Available variables</p> <ul style="list-style-type: none"> <li>index1 string</li> <li>index2 string</li> </ul> | <p>Switch to a different browser session</p> <p>Name*<br/>Switch Browser</p> <p>Description<br/>Switch to a different browser session</p> <p>Input Post Validate</p> <p>Browser window*<br/>\${VARIABLE.index1} (x) ...</p> <p><input type="checkbox"/> Full page screenshot after this action</p> |