

Oracle® Cloud

Using Robots in Oracle Integration 3



F85348-13
February 2026



Oracle Cloud Using Robots in Oracle Integration 3,

F85348-13

Copyright © 2024, 2026, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

About This Content

1 Get Started with Robots

Find More Resources 2

2 Learn About Robots

Start Building Robots Today! 1

Video: Introduction to RPA 1

5 Reasons to Unify Your Automation 2

Create an Automation Strategy 3

About Robots 5

 Robots Automate Without APIs 5

 Differences: Robots and Integrations 7

 When to Create Robots Versus Integrations 7

 Available Robot Patterns 8

 What Happens When a Robot Runs 9

 How Robots, Integrations, and Process Applications Interact 10

 Building Options for Robots 10

 Data Retention for Robots 10

 Security for Robots 11

More Robot Concepts 11

 About Projects 11

 About Robot Connections and Robot Connection Types 12

 Predefined Robot Connection Types 13

 How Robot Connections Work for Multiple Instances 14

 About Environments and Environment Pools 14

 About the Robot Agent 15

 Familiar with the Connectivity Agent? 15

 Your Responsibilities 16

 Plan the Robot Agent Setup 16

 System Requirements 16

 Specifications 17

3 Best Practices and Guidelines

Operations Best Practices	1
Promote Communication	1
Automate Today, Iterate in the Future	1
Plan Strategically Across Teams	1
Understand a Problem Before Solving It	2
Strive for Continuous Improvement	2
Environment Best Practices	2
Set Up Enough Environments	3
Create System Requirements	3
Follow All License Agreements	3
Replicate the Robot's Environment	3
Monitor Your Environments	3
Maintain Good Performance	3
Building and Testing Best Practices	4
Think Like a Robot, Not a Human	4
Think Beyond Robots When Needed	4
Define Inputs Using Robot Connections	4
Use the Wait Action for Slow Applications	4
Use a Foreach Loop for Iterative Work	5
Use Page States to Capture UI Possibilities	5
Future-Proof Your Robots	5
Keep the Canvas Tidy	5
Create Naming Conventions	5
Test in Real-World Conditions	5
Use Screenshots to Help with Testing	6
Processing Bulk Data Best Practices	6
Questions	8
Processing Options	8
Additional Factors: Number of Robots and Records	9
Sample Calculations	10
Simple Scenarios	10
Sequential Processing	11
Parallel Processing	12
Guidelines for Expressions	14
About Expressions	14
Mathematical Operators	14
Comparison Operators	15
Logical Operators	15

Inversion Operators	15
Functions	16
Formatting Rules for Expression Syntax	19
General Formatting Rules	19
Formatting Rules for Placeholder Values	20
When to Convert a String to a Number	22

4 Build a Robot

Complete Prerequisites	1
Review Your Network Configuration	2
Update Your Network's Allowlist	2
When to Update the Allowlist	3
Reasons to Update the Allowlist	3
Update the Allowlist	3
Execute Robot Agents in Air-Gapped Environments	5
Important Key Considerations	6
Export the Robot Agent and Dependencies from the Online Source Environment	7
Import the ZIP File to the Offline Target Environment	8
Create Application Accounts	8
Create a Confidential Application	9
Ensure that the Confidential Application is Active	10
Assign the ServiceDeployer Role to the Confidential Application	11
Meet the Robot Agent's Requirements	12
Download the Robot Agent	12
Update the Robot Agent's Configuration File	13
Install and Start the Robot Agent	15
Install the Robot Agent	15
Start the Robot Agent	16
Determine Success	17
First Time Starting the Robot Agent	18
If the Robot Agent Doesn't Start	18
Stop a Robot Agent	18
Start the Robot Agent Automatically	19
Linux Machines	19
Windows Machines	19
Create a Project	23
Quick Start for Building Robots	23
Workflow for Planning a Robot	24
Identify the Problem	24
Understand the Applications	25
Define the Requirements	25

Workflow for Building a Robot	26
Design Time Robot Statuses	28
Video: Build a Simple Robot in Minutes	28
Video: See a Simple Robot in Action	28
Learn with a Tutorial	29
11 Tips for New Robot Builders	29
Tailor Your Building Experience	32
Create Connections to Applications	34
Create a Robot Connection Type	34
Create a Robot Connection	35
Option 1: Work Outside a Robot	35
Option 2: Work in a Robot	36
Create a Robot and Integration	37
Create a Robot	37
Install the Recorder	39
Design an Integration That Calls a Robot	40
Start from an Accelerator	42
Add an Action to a Robot	42
Add an Alert Action	43
Add a Checkbox Action	46
Add a Clear Text Action	48
Add a Click Element Action	49
Add a Click Element Action Using the Recorder	50
Add a Click Element Action Using the Low-Code Tools	52
Add a Close Browser Action	54
Add a Data Stitch Action	55
Use Cases	55
Add a Data Stitch Action	55
Add a Download File Action	57
Add an Enter Text Action	59
Use Cases	59
Add an Enter Text Action Using the Recorder	59
Add an Enter Text Action Using the Low-Code Tools	61
Add a Frame Action	63
Requirements and Restrictions	63
Add a Frame Action Using the Low-Code Tools	64
Add a Get Text Action	66
Use Cases	66
Add a Get Text Action Using the Recorder	66
Add a Get Text Action Using the Low-Code Tools	68
Add a List Action	70
Use Cases	70

Add a List Action Using the Recorder	70
Add a List Action Using the Low-Code Tools	72
Operations for the List Action	75
Add a Log Action	77
Add an Open Browser Action	78
Add a Radio Button Action	80
Requirements	80
Add a Radio Button Action Using the Recorder	80
Add a Radio Button Action Using the Low-Code Tools	83
Operations for the Radio Button Action	84
Add a Screenshot Action	85
Add a Switch Browser Action	87
Add an Upload File Action	89
Add a Wait Action	92
Add a Wait Until Element Is Visible Action	93
Use Cases	93
Add a Wait Until Element Is Visible Action Using the Recorder	93
Add a Wait Until Element Is Visible Action Using the Low-Code Tools	95
Add a Web Table Action	97
Add a Window Action	100
Deeper Dive: Settings for Robot Actions	104
Define the Fields of an Action	104
Capture Screenshots in Robots	108
Add Validation to a Robot Action	109
Add Logic to a Robot	112
Add a Foreach Loop	112
Add a Switch Condition	114
Add a Stop	117
Fix a Robot's Errors	118
Validate a Robot	119
Fix an Error in a Robot	119
Specify Where a Robot Runs	121
Understand the Rules for Environments and Environment Pools	121
Create an Environment Pool	122
Add Computers to an Environment Pool	123
Associate a Robot with an Environment Pool	123
Update an Environment Pool	124
Remove an Environment Pool from a Robot	125
Create and Update a Robot Resource	125
Alternatives to Hard Coding Data	125
Work with Web Table Definitions	130
Create a Web Table Definition	130

Update a Web Table Definition	132
Work with Variables	132
Create a Variable	132
Update a Variable	133
See Where a Variable Is Used	134
Work with a Trigger's Input and Output	134
Create a Trigger's Input or Output	134
Update a Trigger's Input or Output	135
Work with Page States	136
Create a Page State	136
Update a Page State	138
Work with Targets	138
Create a Target	138
Update a Target	140
See Where a Target Is Used	140
Work with Data Types	141
Create a Data Type	141
Update a Data Type	143
See Where a Data Type Is Used	143
Work with Robot Connections and Robot Connection Types	144
Update a Robot Connection	144
Update a Robot Connection Type	145
View HTML and XPath's	145
View the HTML Code for a Page	145
View an Element's XPath	146
View All Selectors for a Locator Field	147

5 Run and Test a Robot

Workflow for Testing a Robot	1
Open a Robot	2
Activate a Robot	2
Tracing Levels for Robots	4
Log Screenshots to Activity Stream	4
Options for Tracing Levels	5
Test a Robot on Its Environment	6
Sample Code for Testing a Robot	7
Test a Robot and Its Integration	9
Activate an Integration	9
Run an Integration and Robot	9

6 Deploy and Monitor a Robot

Deploy a Robot	1
Workflow for Deploying a Robot	1
Deploy a Robot to a Higher Environment	2
Monitor an Automation Solution	3
Monitor a Robot	3
Monitor a Robot and Its Integration	6
Monitor Environments and Environment Pools	9
Check Whether an Environment is Available	13
Runtime Statuses of Robot Instances	14
Update a Deployed Robot	14
Workflow for Updating a Deployed Robot	15
Version a Robot	15
Replace a Robot with an Integration	16

7 Troubleshoot

Troubleshoot the Robot Agent and Environments	1
A Robot Agent Doesn't Start	1
An Environment Is Unavailable	8
An Unused Environment Still Appears	9
Troubleshoot the Recorder	10
The Recorder Doesn't Start	10
The Recorder Stops Working	11
An Application Isn't in the Browsers List	12
The Canvas Is Read-Only	13
The Recorder Doesn't Capture the Correct Element for an Image	13
An Error Occurs When Targeting an Inline Frame	15
Troubleshoot Robots	17
A Robot Doesn't Activate	17
A Robot Fails	17
Robot Page Is Blank in a Project	20
Download the Log File for a Robot or Robot Agent	21

A Use Cases

Use Case: Update a Set of Invoices	A-1
Use Case: Save Data After Iterating on Invoices	A-10
Use Case: Save Values After Iterating on a Table	A-15
Use Case: Switch Browsers	A-25

About This Content

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

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 <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

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.

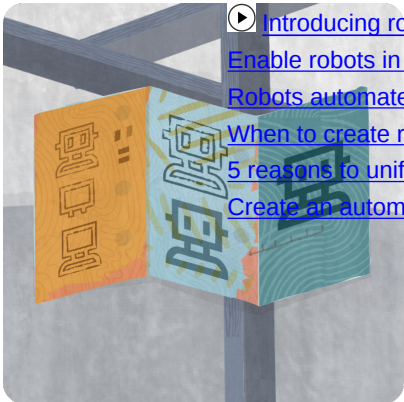
Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type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

1

Get Started with Robots

A robot completes a task by working in an application, just like a person does. Browse the following popular topics to learn more.


Explore Robots

Image	Links
	<ul style="list-style-type: none">▶ Introducing robotic process automation (RPA)Enable robots in your instanceRobots automate without APIsWhen to create robots versus integrations5 reasons to unify your automationCreate an automation strategy

Build and Test a Robot

Image	Links
	<ul style="list-style-type: none">▶ Part 1: How to build a simple robot▶ Part 2: See a simple robot in actionComplete prerequisites: Download the robot agent Start the robot agentWorkflows: Plan a robot Build a robot Test a robot Deploy a robotExplore real-world use cases

Troubleshoot

Image	Links
	Troubleshoot the robot agent and environments Troubleshoot the recorder Troubleshoot robots

Find More Resources

Oracle Integration offers many ways to stay connected and grow your expertise.

Resource	Description	Link
Documentation	Find step-by-step instructions, best practices, and more.	Get Started page on the Oracle Help Center
Tutorials	Take a product tour, build an integration from start to finish, or complete a longer self-directed workshop with a LiveLab.	Tours and Tutorials in <i>Using Integrations in Oracle Integration 3</i>
Videos	Learn more about Oracle Integration by watching a short video.	Videos page on the Oracle Help Center
Live demos	Created by the product management team, live demos offer deep dives into new features, use cases, and more. Subscribe to the channel to get notified when a new live demo is available.	Oracle Integration on YouTube
Oracle Integration blog	Learn what's new from the product management team.	Oracle Integration blog
A-Team Blog	Learn from technical solution architects and software engineers.	A-Team Chronicles blog
Cloud Customer Connect	Connect with experts, ask or answer questions, and share your opinions.	Cloud Customer Connect
Newsletter	Oracle Integration sends a quarterly newsletter with product, event, and training information.	Subscribe to the Oracle Integration Newsletter
Webcasts	Oracle Integration offers the following quarterly webcasts: <ul style="list-style-type: none"> Product Update Webcasts Get product updates and watch live demos. Customer Success Webcasts Get inspired by success stories and best practices, or present your own wins to the community. 	Integration Resources blog post: Watch past webcasts and learn about upcoming webcasts Subscribe to the Oracle Integration Newsletter : Learn about upcoming webcasts by subscribing to the newsletter

Resource	Description	Link
Oracle Partners	Use the Oracle PartnerNetwork to find a partner to support your business goals. Connect to the Oracle network to view this page.	Oracle Partner Finder

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	Video: Introduction to RPA Video: Build a Simple Robot in Minutes Video: See a Simple Robot in Action
Plan how to fit robots into your business	5 Reasons to Unify Your Automation Create an Automation Strategy
Learn about robots and robotic process automation (RPA)	Start Building Robots Today! About Robots More Robot Concepts

Start Building Robots Today!

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

- **Already using Oracle Integration?**
 - If robotic process automation is available in your region, you can start building robots today. To see the regions in which RPA is available, see Availability in *Provisioning and Administering Oracle Integration 3*.
 - If robotic process automation isn't available in your region yet, enter a service request (SR) so that RPA can be enabled.

- **Not yet using 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.

Video: Introduction to RPA

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



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](#).

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 business 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, you can unify your automation work and your 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 perform UI-based automation and API-based automation in one place—Oracle Integration—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: a project.

Whether you're developing a new robot or integration, or updating an existing robot or integration, you work in a project. 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:

- You don't have a central place to monitor all automation. Instead, 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 to 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 API-based actions or a new integration.

Learn More

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

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](#).

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 example, 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 lifecycle 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](#)
- [Data Retention for Robots](#)
- [Security 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 much in common. 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 in a project.
- Similar observability pages are available 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, a robot might need to 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 requires a little 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 completes API-based automation tasks instead of UI-based automation tasks.

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	Integration	Robot
How it gets information from an application	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.
Key requirements	An integration developer with a broad and deep technical skill set designs an integration. The integration must access the application's APIs.	A robot developer who is comfortable with technology builds a robot. The robot must access an application's user interface.
Sample usage for an ERP system	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 for any of the following scenarios:

1. No APIs

The applications that you're automating have no APIs; or their APIs can't access and update the fields that the business process uses.

2. Operational and logistical challenges

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 replace a robot with an integration without impacting your business process. See [Replace a Robot with an Integration](#).

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	You can build a robot that interacts with any HTML-based web application, such as a website or an application that you access from an internet browser.	<p>Balancing invoices</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 tedious and manual work.</p> <p>Getting the supplier name from an invoice</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 task.</p> <p>Updating emergency contact information</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. Next, Oracle Integration looks within the environment pool and selects an environment that is available to run the robot.
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. Then, the robot agent 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.

Order	Task	Details
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 example, 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 always use your tool of choice to build. You can start and stop the recorder from anywhere in your robot. Additionally, you can switch from the recorder to the low-code tools at any time.

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](#).

Data Retention for Robots

Robot data is retained according to the data retention period of an Oracle Integration service instance.

See [Edit the Data Retention Period for an Instance in Provisioning and Administering Oracle Integration 3](#).

Security for Robots

Understand how to secure robots and their credentials.

Sign in to Applications with Multifactor Authentication (MFA)

Some web applications require multifactor authentication (MFA), also known as two-factor authentication (2FA). MFA requires users to verify their identify in multiple ways. Robots frequently encounter difficulties progressing beyond MFA authentication.

Here's the recommended approach:

- When an application requires MFA, use a service account for the robot.
A service account is used only by applications and automated processes.
- Disable MFA for the service account.

Store Application Credentials

A robot requires credentials to access an application, and you need to store the credentials in a safe location.

The recommended approach is to store the credentials using a security service, such as OCI Vault. See [Use Vault Secret Credentials with Oracle Cloud Infrastructure Vault](#) in the Oracle Cloud Infrastructure Documentation.

Find More Security Information

A security guide is available for Oracle Integration. See Overview of Oracle Integration Security in *Securing Oracle Integration 3*.

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

Build your automation solutions, manage their deployment, and observe them in a project.

New to Projects?

A project is the hub of all your automation work. It's also 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 achieve 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:

- You create and manage the following components in a project:
 - Robots
 - Robot connections and robot connection types
 - Environment pools
 - Integrations that call robots
- 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
Basis of the connection	An integration connection is based on an adapter. 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 type, or create your own. A robot connection type defines how to connect to an application, such as by specifying a URL and user credentials.
Connection method	An integration connection uses Java code to call an API. 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 uses to open an application.
Testing	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, robot connections to different applications can be based on the same robot

connection type. For example, if a robot requires a user name, password, and URL to access an application, you can base its robot connection on the predefined robot connection type. See [Predefined Robot Connection Types](#).

If an application requires parameters that aren't included in the predefined robot connection type, 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 deactivating it.
Therefore, you can quickly update a robot when credentials change.

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.

The following predefined robot connection type is available.

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

If your application requires different parameters, you can base a robot connection on a robot connection type that you create. See [Create a Robot Connection Type](#).

How Robot Connections Work for Multiple Instances

You might have multiple service 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	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. 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. 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. 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. Identify any differences between the development and production instances of the application. These differences could impact the robot's behavior in production. For example, 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.

For details about deployment, see [Workflow for Deploying a Robot](#).

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 a robot's duration is the key to setting up environments effectively and efficiently.

For example, 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.

Flexibility Thanks to Shared Environment Pools

You create an environment pool in a project. You have several options for the robots that the environment pool can run:

- Only robots in the project
- Robots in all projects

This flexibility allows you to organize automation solutions in a way that works for your business, while still having the flexibility to choose the right environment pool for each robot to run on. See [Create an Environment Pool](#).

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.
See [Specify Where a Robot Runs](#)
- 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.
See [Add Computers to an Environment Pool](#).
- Installing and managing the robot agent on each environment.
See [About the Robot Agent](#).

About the Robot Agent

Install one or more robot agents on each environment. An environment is the computer or virtual machine where a robot runs. A robot agent runs the robot instances on its environment.

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
Usage	Use the connectivity agent to connect to a resource on a private or on-premises network from an integration.	Use the robot agent to run a robot.
Installation and number	Install the connectivity agent on an on-premises machine, typically a virtual machine. Your organization can install one or many connectivity agents.	Install the robot agent on every environment that must run a robot. You can install one or more robot agents per environment, and you can create as many environments as you need.
Components	Two types of connectivity agents are available: <ul style="list-style-type: none"> SaaS agent, which runs in Oracle Integration. Oracle manages this agent. On-premises agent, which you must install and manage. 	Only one robot agent is available.

Your Responsibilities

Your organization is responsible for several tasks.

- Plan the robot agent setup for your organization.
See [Plan the Robot Agent Setup](#).
- Ensure that all environments meet the system requirements for the robot agent.
See [System Requirements](#).
- Install the robot agent on every environment and, if required, configure the robot agent as a Windows service.
See [Design Considerations](#) and [Complete Prerequisites](#).
- Keep the robot agent up to date on every environment.

Plan the Robot Agent Setup

When planning your robot agent setup, consider the number of robots agents that you need in total, the number of robot agents to install on each environment, and the number environments to create.

For more guidance, see [Set Up Enough Environments](#).

System Requirements

Every environment must meet the system requirements. The following system requirements are for an environment that runs a single robot agent.

Area	Requirements
Hardware	Minimum of 8 GB memory with 4 GB of heap size dedicated to the robot agent If you run other processes on the environment besides the robot agent, Oracle recommends increasing the physical memory

Area	Requirements
Operating system	<p>Install the robot agent on one of the following certified operating systems:</p> <ul style="list-style-type: none"> • MacOS • Oracle Linux 6.x, 7.x, and 8.x • RedHat Enterprise Linux 6.6, 7.x, and 8.x • Suse Linux Enterprise Edition 12 SP2 • Windows Standard Edition 2016 • Windows 2019
Software	<ul style="list-style-type: none"> • Java Development Kit (JDK) 17 <p>You do not need to purchase a JDK license for the sole purpose of installing and/or running the connectivity agent. See JDK license and support for OIC Connectivity Agent (Doc ID 2611142.1).</p> <p>The JDK installation can be the same one as used with other products installed on the same host. However, separate JDK licensing may be required for any use of Java beyond the use of the connectivity agent. Ensure that the JDK installation is <i>not</i> modified for use with these other products.</p> • One of the following browsers for the environment where the robot agent runs: <ul style="list-style-type: none"> – Apple Safari – Google Chrome (recommended) or Headless Chrome – Microsoft Internet Explorer – Microsoft Edge – Mozilla Firefox or Headless Firefox <p>Note: Oracle Integration has different browser requirements. See Supported Browsers in <i>What's New for Oracle Integration 3</i>.</p> • One of the following options for storing secret information in the computer's local keystore: <ul style="list-style-type: none"> – Linux and Unix computers: Keyutils <p>You can install Keyutils by installing Homebrew, Yum, or AptGet.</p> – Windows computers: PowerShell <p>This solution is usually included in the operating system of Windows computers.</p> – Mac computers: Security Utility <p>This software is usually included in the operating system of Mac computers.</p>

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"> • The robot agent initiates all communication and must be able to contact Oracle Integration so that it can poll for work. See Review Your Network Configuration. • No ports are opened on the on-premises system for communication.

Area	More information
Security protocol	<ul style="list-style-type: none"> The robot agent registers with Oracle Integration over SSL using the provided Oracle Integration credentials. See Complete Prerequisites. All communication is secured using SSL. The robot agent processes requests by pulling messages from Oracle Integration across SSL. The robot agent posts responses by pushing messages to Oracle Integration across SSL.
Data persistence	No data is persisted in the robot agent.

Design Considerations

A robot runs on an environment, which is a computer or virtual machine (VM) on which the robot agent is installed. When determining where to place the environment, consider the requirements for the robot's interaction.

Type of robot interaction	Location for the environment
Interact with a publicly available website	<p>Ensure that the robot's environments are within a network that can interact with a publicly available web application.</p> <p>If the computer that hosts the robot agent has restrictions on outbound connections to the public internet, you might need to update your network's allowlist to allow outbound access to the websites that are required for the robot to run. See Update Your Network's Allowlist.</p>
Interact with an internal web application that is within a firewall and VPN	<p>Ensure that the robot's environments are within the VPN with the HTTPS proxy settings configured so that the environments have access to the external web applications.</p> <p>The environment must be able to access the public internet the first time that you start the robot agent and the first time that you run a robot that launches an internet browser. The robot agent requires this access to download its required dependencies. Therefore, you might need to temporarily disable egress restrictions on the environment the first time you start the robot agent.</p>
Interact with a VM setup that has no public internet access (air-gapped environments)	You can export robots and their complete runtime dependencies from a source environment and import them into a target environment for fully offline execution . See Execute Robot Agents in Air-Gapped Environments .

3

Best Practices and Guidelines

Spend some time reviewing the best practices 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, teams often divide automation responsibilities.

- One person or team builds a robot.
- Another person or team designs the integration that calls the robot.
- A third person or team monitors 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 might become what your software can support, rather than what your business requires, and it might solve 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 in the application that the robot automates.
- 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

Most computers have 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 Environments

Periodically throughout the day, monitor your environments. For example, ensure that all environments remain up and running.

See [Check Whether an Environment is Available](#).

Maintain Good Performance

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

Scenario	How to plan for the scenario
A robot doesn't understand an element in a user interface (UI)	Remember that a robot interacts with an application differently than a human does. Build your robot intentionally for the robot's requirements. For example, many people 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	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.

Think Beyond Robots When Needed

Robots can automate many of your business processes, but remember that robots are just one of many automation technologies that are available to you in Oracle Integration. Choose the right technology for the goal you're trying to achieve.

For example:

- If you need to insert information into a database or work with an application's APIs in another way, use an integration.
- If you need to automate decision making, use a decision.

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.

Use the Wait Action for Slow Applications

When the load time for a web page is slower than expected, a robot sometimes can't complete all of its work. For example, the robot might be able to enter data into all the fields but doesn't have time to capture the entered data before the application closes.

The wait action helps a robot work around application latency by pausing the robot's actions and giving the application time to catch up. See [Add a Wait Action](#).

Use a Foreach Loop for Iterative Work

A robot signs in to an application at the beginning of its flow and signs out of an application after it completes its work. However, when a robot must perform many transactions, these repeated sign-ins and sign-outs can impact the performance of your business process.

When a robot must perform work on multiple business objects, allow the robot to iterate over them by using a foreach loop. See [Add a Foreach Loop](#).

Use Page States to Capture UI Possibilities

When a user interface might present several conditions to a robot, use page states to handle the conditions.

To handle each page state differently, use the page in a switch condition. See [Work with Page States](#) and [Add a Switch Condition](#).

Future-Proof Your Robots

Don't hard-code data unless you 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.

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.

The names of some objects, such as robots, robot connections, and environment pools, can include spaces. Other more programmatic objects, such as variables and input parameters, cannot include spaces in their names.

If your organization doesn't have them yet, create naming conventions. Then, follow the conventions so that everyone understands the components at a glance and can 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

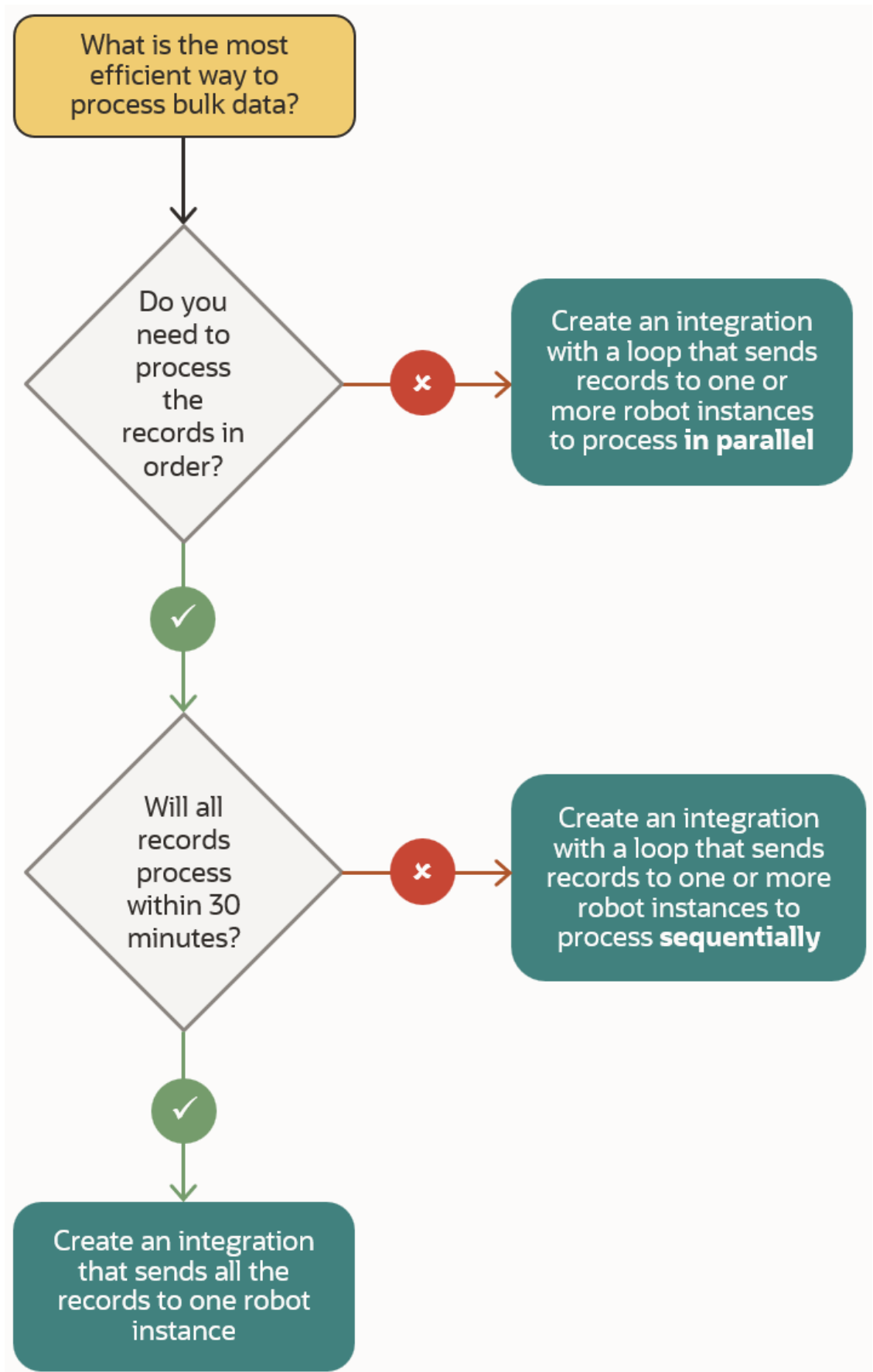
When you add an action to a robot, you have several options for capturing screenshots, including capturing a robot's view before and after completing an action. The screenshots are helpful when troubleshooting a failed robot instance.

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 in the user interface.

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 example, 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>1</p> <p>Create an integration with a loop that sends records to one or more robot instances to process in parallel</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>
<p>2</p> <p>Create an integration with a loop that sends records to one or more robot instances to process sequentially</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"> • Each record takes a long time to process. • You want to include error handling. <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>3</p> <p>Create an integration that sends all the records to one robot instance to process sequentially</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>

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:

<p>1</p> <p>Create an integration with a loop that sends records to one or more robot instances to process in parallel</p>	<p>2</p> <p>Create an integration with a loop that sends records to one or more robot instances to process sequentially</p>
---	--

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 (1m 15s) 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"> • Number of records to process • Overhead for the robot • Number of robots that process records <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 (1m 15s) 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 1m 15s to run, resulting in a total processing time of 6m 25s:</p> $1\text{m }15\text{s processing time} \times 5\text{ robots} = 6\text{m }25\text{s processing time}$ <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 2m 25s:</p> $1\text{ minute overhead} + (15\text{ seconds processing time per record} \times 5\text{ records}) = 2\text{m }15\text{s processing time}$

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 example, if one robot updates 150 records, you save 149 minutes of overhead time. However, the total processing time is 38m 30s, which might be longer than you want to wait to determine whether all the updates completed successfully.</p> <p>1 minute overhead + (15 seconds processing time per record x 150 records) = 38m 30s processing time</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 example, 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 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:

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

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:

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

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

Scenario	Total processing time per branch	Calculation
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) 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) 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) This value exceeds the 30-minute time limit	60 seconds of overhead + (125 records x 15 seconds of processing time) = 1935 seconds

Scenario	Total processing time per branch	Calculation
4 branches, 125 records per branch, send in 2 different batches	Each batch finishes in 1005 seconds (16 $\frac{3}{4}$ minutes), for a total processing time of 2010 seconds (33 $\frac{1}{2}$ 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 $\frac{1}{2}$ 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

Operator	Python equivalent	Description	Expression	Returns
-	-	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	$\{1==1\}$ $\{one==one\}$	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	$\{1<=2 \ \&\& \ 2<=3\}$	true
	or	Either statement must be true	$\{2<=1 \ \ 3<=2\}$	false

Inversion Operators

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

Operator	Python equivalent	Description	Expression
!	not	NOT operation, which inverses the result of a comparison Use parentheses to clearly separate the NOT operation (!) from the expression.	$\{!(1>=2)\}$ returns true $\{!(1<=2 \ \&\& \ 2<=3)\}$ returns false $\{!(2<=1 \ \ 3<=2)\}$ returns true

Functions

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

Function	Description	Format	Example
<code>count()</code>	Returns the count of a collection of elements. Python equivalent: <code>count()</code>	$\${count(value)}$ where: <i>value</i> is the name of a collection variable	Return the count of a numeric collection variable $\${count(\$VARIABLE.numberCollection1)}$ where: numberCollection1 is the name of a collection variable
<code>first()</code>	Returns the first element in a collection. Python equivalent: <code>first()</code>	$\${first(value)}$ where: <i>value</i> is the name of a collection variable	Return the first element of a numeric collection variable $\${first(\$VARIABLE.numberCollection1)}$ where: numberCollection1 is the name of a collection variable
<code>last()</code>	Returns the last element in a collection. Python equivalent: <code>last()</code>	$\${last(value)}$ where: <i>value</i> is the name of a collection variable	Return the last element of a numeric collection variable $\${last(\$VARIABLE.numberCollection1)}$ where: numberCollection1 is the name of a collection variable
<code>toNumber()</code>	Returns an integer or float for a numerical string. Python equivalent: No direct equivalent, though you can use <code>int()</code> and <code>float()</code> to convert strings to numbers	$\${toNumber(value)}$ where: <i>value</i> is the number to return	Return a number from a numeric string value <code>toNumber("100.00")</code> returns 100.00 <code>toNumber("100,000")</code> returns 100000 <code>toNumber("100,000.00")</code> returns 100000.00 <code>toNumber(\$VARIABLE.amount)</code> returns a value according to the value of the variable: <ul style="list-style-type: none"> If <code>\$VARIABLE.amount</code> is 100, returns 100 If <code>\$VARIABLE.amount</code> is 100.00, returns 100.00 If the string value contains any non-numeric characters, an error occurs. For example, <code>toNumber("100A")</code> returns an error

Function	Description	Format	Example
toString()	Returns a string that represents an object. Python equivalent: str()	<code>\${toString(value)}</code> where: <i>value</i> is the string to return	Return a string that represents a number <code>\${toString(1)}</code> returns 1 Return a person's first and last name You define a data type named <i>Person</i> . <i>Person</i> contains two properties: <i>First_name</i> and <i>Last_name</i> . Next, you define a variable, <i>var1</i> . Base <i>var1</i> on the <i>Person</i> data type. A robot returns the following values for the variable: <ul style="list-style-type: none"> <code>var1.FirstName = "John"</code> <code>var1.Lastname = "Doe"</code> Given this scenario, the following expression: <code>\${toString(var1)}</code> returns { "FirstName": "John", .. "lastName": "Doe" }
trim()	Remove leading and trailing white space characters, including spaces, tabs, and other whitespace at the beginning and end of text.	<code>\${trim(value)}</code> where: <i>value</i> is the name of a variable	Return a value with no leading or trailing white space characters <code> \${trim(\$VARIABLE.company_name)}</code> where <code>company_name = ' Oracle '</code> returns Oracle
subString()	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 <code>subString</code> function for the value ABC: <ul style="list-style-type: none"> A is entry 0 B is entry 1 C is entry 2 Python equivalent: <code>\$name[start:stop]</code>	<code>\${subString(string, startIndex, endIndex)}</code> where: <ul style="list-style-type: none"> <i>string</i> is the string to evaluate <i>startIndex</i> is the first character to include in the returned substring <i>endIndex</i> is the last character to include in the returned substring 	Return a subset of letters <code>\${subString(ABCDEF, 0, 0)}</code> returns A <code>\${subString(ABCDEF, 1, 1)}</code> returns B <code>\${subString(ABCDEF, 2, 4)}</code> returns CDE Return the last four digits of a phone number <code>\${subString(5551234567, 6, 9)}</code> returns 4567

Function	Description	Format	Example
formatString()	<p>Formats a template string using positional placeholders.</p> <p>Converts the template and all provided values to string, then applies string formatting with positional arguments only. This supports placeholders like {}, {0}, {1}.</p>	<pre> \${VARIABLE.result} = \$ {formatString(\$VARIABLE.template, \$VARIABLE.values)} </pre> <p>where:</p> <ul style="list-style-type: none"> <i>template</i> is the template string to be formatted <i>values</i> are the collection of values to substitute during runtime 	<p>Return the formatted string for the following examples</p> <p>Basic positional formatting</p> <pre> \${formatString("Hello, {0}!", ["Alice"])} </pre> <p>returns</p> <pre> "Hello, Alice!" </pre> <p>Multiple values</p> <pre> \${formatString("{} has {} new messages", ["Bob", "5"])} </pre> <p>returns</p> <pre> "Bob has 5 new messages" </pre> <p>Indexed placeholders</p> <pre> \${formatString("{1}, {0}, and {1} again", ["first", "second"])} </pre> <p>returns</p> <pre> "second, first, and second again" </pre> <p>Extra values are ignored</p> <pre> \${formatString("Only {} is used", ["this", "that"])} </pre> <p>returns</p> <pre> "Only this is used" </pre> <p>Missing values raise IndexError</p> <pre> \${formatString("{} {}", ["one"])} </pre> <p>returns</p> <pre> IndexError due to missing argument values. </pre> <p>Named placeholders raise KeyError</p> <pre> \${formatString("User: {name}", ["Alice"])} </pre> <p>returns</p> <pre> KeyError because keyword arguments are not allowed. </pre> <p>Escaping braces</p> <pre> \${formatString("Set is {{1, 2, {}}}", [3])} </pre> <p>returns</p> <pre> "Set is {1, 2, 3}" </pre> <p>Custom XPath expression</p> <pre> \${formatString("xpath://div[normalize-space(text()) = '{}']//div[normalize-space(text()) = '{}', {}]/..", ["PERTH", "TAS", "7300"])} </pre>

Note
Named placeholders are not supported in XPath expressions.

Function	Description	Format	Example
	<p>Python equivalent: <code>str(template).format(*(str(v) for v in values))</code></p>	<pre>not supported.</pre>	<p>returns</p> <pre>"xpath://div[normalize-space(text()) = 'PERTH']//div[normalize-space(text()) = 'TAS,7300']/.."</pre> <p>For the Data Stitch action:</p> <ul style="list-style-type: none"> Prepare the template as follows: <pre> \${VARIABLE.template} = "xpath://div[normalize-space(text()) = '{ }']//div[normalize-space(text()) = '{ },{ }']/.." </pre> Prepare the values collection as follows: <div style="border: 1px solid #ccc; border-radius: 10px; padding: 10px; margin: 10px 0;"> <p>Note</p> <p>+ indicates an Append operation.</p> </div> <pre> \${VARIABLE.values} + "PERTH" \${VARIABLE.values} + "TAS" \${VARIABLE.values} + \${toString(7300)} Final values, ["PERTH", "TAS", "7300"] </pre> <p>For a data string, use the <code>formatString</code> function as follows: <pre> \${VARIABLE.result} = \$ {formatString(\${VARIABLE.template}, \${VARIABLE.values})} </pre> </p>

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: <pre> \${toString(\$VARIABLE.CurrentInvoice[InvoiceAmount]) == \$VARIABLE.InvAmt && \$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: <pre> \${ "https://www.mycompany.com/" } </pre>
You can use literal numbers but not exponents	Supported values: <ul style="list-style-type: none"> • 1, -1, and +1 • 1.56, -1.56 and +1.56 Not supported: <ul style="list-style-type: none"> • 1e2 • 2^2
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.

Formatting Rules for Placeholder Values

Each placeholder value, such as a variable, robot connection, or input and output property, has its own formatting rules for expression syntax.

Value	Format
 Input properties	<p>Format for an input property</p> <pre> \${INPUT.input_property_name} </pre> <p>where:</p> <ul style="list-style-type: none"> • <i>input_property_name</i> is the name of the input property. <p>Format for a data type property on an input property</p> <p>Use this format when you base an input 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"> • Option 1: <code>\${INPUT.input_property_name[property_name]}</code> This format allows you to reference a property name that includes spaces in its value. • Option 2: <code>\${INPUT.input_property_name.property_name}</code> This format doesn't allow spaces 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. <p>where:</p> <ul style="list-style-type: none"> • <i>input_property_name</i> is the name of the input property. • <i>property_name</i> is the name of the data type property that the input property is based upon. <p>Note: Inputs properties can be nested. Follow the same formatting rules for nested properties. For example:</p> <ul style="list-style-type: none"> • <code>\${INPUT.input_property_name[property_name][property_name][property_name]}</code> • <code> <pre> \${ INPUT.input_property_name.property_name.property_name.prope rty_name} </pre> </code>

Value	Format
 Output properties	<p>Format for an output property</p> <pre>#{OUTPUT.output_property_name}</pre> <p>where:</p> <ul style="list-style-type: none"> <i>output_property_name</i> is the name of the output property. <p>Format for a data type property on an output property</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"> Option 1: <pre>#{OUTPUT.output_property_name[property_name]}</pre> This format allows you to reference a property name that includes spaces in its value. Option 2: <pre>#{OUTPUT.output_property_name.property_name}</pre> This format doesn't allow spaces 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. <p>where:</p> <ul style="list-style-type: none"> <i>output_property_name</i> is the name of the property of the output. <i>property_name</i> is the name of the data type property that the output property is based upon. <p>Note: Output properties can be nested. Follow the same formatting rules for these nested properties. For example:</p> <ul style="list-style-type: none"> <pre>#{OUTPUT.output_property_name[property_name][property_name][property_name]}</pre> <pre>#{OUTPUT.output_property_name.property_name.property_name.property_name}</pre>
 Page states	<pre>#{pageState("page_state_name")}</pre> <p>where:</p> <ul style="list-style-type: none"> <i>page_state_name</i> is the name of the page state
 Robot connections	<pre>#{CONNECTION.robot_connection_name.property}</pre> <p>where:</p> <ul style="list-style-type: none"> <i>robot_connection_name</i> is the name of the robot connection <i>property</i> is the name of the property in the robot connection
 Targets	<pre>#{TARGET.target_name}</pre> <p>where:</p> <ul style="list-style-type: none"> <i>target_name</i> is the name of the target.

Value	Format
(x) Variables	<p>Format for a variable</p> <pre>#{VARIABLE.variable_name}</pre> <p>where:</p> <ul style="list-style-type: none"> <i>variable_name</i> is the name of the variable. <p>Format for a data type property on a variable</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"> Option 1: <pre>#{VARIABLE.variable_name[property_name]}</pre> This format allows you to reference a property name that includes spaces in its value. Option 2: <pre>#{VARIABLE.variable_name.property_name}</pre> This format doesn't allow spaces 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. <p>where:</p> <ul style="list-style-type: none"> <i>variable_name</i> is the name of the variable. <i>property_name</i> is the name of the property of the data type that the variable is based upon. <p>Note: Variables can have nested properties. Follow the same formatting rules for these nested properties. For example:</p> <ul style="list-style-type: none"> <pre>#{VARIABLE.variable_name[property_name][property_name][property_name]}</pre> <pre>#{VARIABLE.variable_name.property_name.property_name.property_name}</pre>

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 start, and then get started. Don't forget to complete the prerequisite tasks first.

What Do You Want to Do?

Goal	More information
Get started	Quick Start for Building Robots
Build a robot and its related components	<ol style="list-style-type: none">1. Complete Prerequisites2. Create Connections to Applications3. Create a Robot and Integration4. Add an Action to a Robot5. Add Logic to a Robot6. Fix a Robot's Errors7. Specify Where a Robot Runs
Create the robot resources that offer alternatives to hard-coding data	Create and Update a Robot Resource
Get help with the underlying HTML for a page or UI element	View HTML and XPath

Complete Prerequisites

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

Task	More information
Review your network configuration	<ol style="list-style-type: none">1. Review Your Network Configuration2. Update Your Network's Allowlist (Required only if your network has restrictions on outbound connections)3. Execute Robot Agents in Air-Gapped Environments (Required only if your network has no public internet access)
Create accounts for robots and people	<ol style="list-style-type: none">3. Create Application Accounts
Create a confidential application	<ol style="list-style-type: none">4. Create a Confidential Application (Optional)5. Ensure that the Confidential Application is Active (Required only for troubleshooting)6. Assign the ServiceDeployer Role to the Confidential Application (Required only if you created a confidential application)

Task	More information
Install and configure the robot agent	7. Meet the Robot Agent's Requirements 8. Download the Robot Agent 9. Update the Robot Agent's Configuration File 10. Install and Start the Robot Agent 11. Start the Robot Agent Automatically (Optional)
Create a project	12. Create a Project

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. Additionally, identify the correct network location for environments.

The robot agents need internet access for proper execution because they rely on several external services and components. When a robot agent starts for the first time, it will download the required dependencies for execution from the public internet.

1. Review your organization's network configuration. Ensure that the robot agent can poll for work by contacting Oracle Integration.

For example, 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](#).
2. **Restricted internet access:** If the computer that hosts the robot agent has restrictions on outbound connections to the public internet, you might need to update your network's allowlist to allow outbound access to the websites that are required for the robot to run. See [Update Your Network's Allowlist](#).
3. **No internet access:** If you have a secure VM setup in an air-gapped environment with no internet access, you can export the robots and their dependencies from an online source environment as a ZIP file. You can then import this ZIP file into an offline target environment and execute robot agents without internet access.. See [Execute Robot Agents in Air-Gapped Environments](#)
4. Determine the appropriate network location for the environments that host the robot agent. See [Design Considerations](#).

Update Your Network's Allowlist

If the computer that hosts the robot agent has restrictions on outbound connections to the public internet, you might need to update your network's allowlist to allow outbound access to the websites that are required for the robot to run.

Note

You can run the `generateEgressAllowlist` script to resolve the provided domain and any associated CNAMEs into their IPv4 addresses. The script generates the deduplicated list of allowed egress IPs. It is useful for firewall or network whitelisting where outbound access needs to be tightly controlled.

- **Linux:** `[AGENT_HOME]/scripts/generateEgressAllowlist.sh`
- **Windows (PowerShell):** `[AGENT_HOME]/scripts/generateEgressAllowlist.ps1`

When to Update the Allowlist

Updating your network's allowlist is not required when the robot agent's computer has unrestricted access to and from the public internet.

However, if your network restricts outbound connections to the public internet, you probably need to update the allowlist. For example, update the allowlist if either of the following statements is true:

- Your organization installed the robot agent on a computer in a private subnet, which doesn't have direct access to or from the public internet.
- Your organization restricts the outbound connections from the robot agent's VM to specific websites.

Reasons to Update the Allowlist

Updating the allowlist allows for improved security, compliance, and cost and performance control.

- **Security:** Restrict outbound traffic to only trusted destinations, and prevent unauthorized data exfiltration.
- **Compliance:** Meet regulatory requirements by limiting traffic to only approved networks and services, and avoid the accidental exposure of sensitive data.
- **Cost and performance control:** Prevent unnecessary or malicious outbound traffic, and keep workloads focused on intended services (for example, allowing access to only OCI Object Storage or specific corporate networks).

Update the Allowlist

Update your network's allowlist to allow outbound access to robots' required websites.

1. Update the allowlist for your corporate firewall so that you allow outbound calls to the required websites. You have several options:
 - **Option 1 (recommended):** Add domain names to the allowlist. See [Option 1. Domain Names](#).
 - **Option 2:** Add CIDR ranges to the allowlist. See [Option 2. CIDR Ranges](#).

2. If you add CIDR ranges, update your work instructions so that you periodically get the latest IP addresses and CIDR ranges for the websites in the table.

For example, use command-line tools such as **dig** (Domain Information Groper) or **nslookup** (Name Server Lookup) to query Domain Name System (DNS) servers for the IP address(es) associated with the website's domain name. Then, update the CIDR ranges in your allowlist.

3. Update your work instructions so that everyone who builds robots periodically updates their internet browser and WebDriver versions.

For example, use the following URL to download these components for the Google Chrome browser:

<https://googlechromelabs.github.io/chrome-for-testing/#stable>

Option 1. Domain Names

The recommended approach is adding the following domain names to your network's allowlist.

- Required for everyone:
 - `pypi.org`
 - `files.pythonhosted.org`
 - `conda-forge.org`
 - `anaconda.org`
 - `github.com`
- Required if the robot works in Google Chrome:
 - `storage.googleapis.com`
- Required if the robot works in Firefox or GeckoDriver:
 - `objects.githubusercontent.com`
- Required if the robot works in Microsoft Edge:
 - `msedgewebdriverstorage.blob.core.windows.net`

Option 2. CIDR Ranges

Adding CIDR ranges to your network's allowlist is an option but isn't the recommended approach. IP addresses and CIDR ranges can change over time, so an IP address that is valid today could be invalid tomorrow.

Website	CIDR range
Oracle Integration endpoint	Obtain this value from the <code>InstallerProfile.cfg</code> file, which is one of the files that you download for installing the robot agent. See Download the Robot Agent .
Oracle Cloud Infrastructure Identity and Access Management (IAM) endpoint	Obtain this value from the <code>InstallerProfile.cfg</code> file, which is one of the files that you download for installing the robot agent. See Download the Robot Agent .

Website	CIDR range
Websites that robots work in	Obtain these values on your own.
conda.anaconda.org	104.19.144.37/32 104.19.145.37/32
conda-forge.org	104.21.48.1/32 104.21.32.1/32 104.21.16.1/32 104.21.96.1/32 104.21.80.1/32 104.21.112.1/32 104.21.64.1/32
api.anaconda.org	104.19.144.37/32 104.19.145.37/32
pypi.org	151.101.192.223/32 151.101.128.223/32 151.101.0.223/32 151.101.64.223/32
downloads.robotorp.com	104.20.45.65/32 172.66.163.236/32
files.pythonhosted.org	This set of IP addresses is different for each region's Content Delivery Network (CDN). To obtain the IP addresses for your region, run either of the following commands on the computer where you installed the robot agent: <ul style="list-style-type: none"> "dig files.pythonhosted.org +short" "nslookup files.pythonhosted.org"
Google's DNS endpoint	8.8.8.8/32

Execute Robot Agents in Air-Gapped Environments

If you have a secure VM setup in an air-gapped environment with no internet access (offline environment) and want to execute robot agents without internet access, you can export the robots and their dependencies from an online source environment as a ZIP file. You can then import this ZIP file into an offline target environment and execute robot agents without internet access.

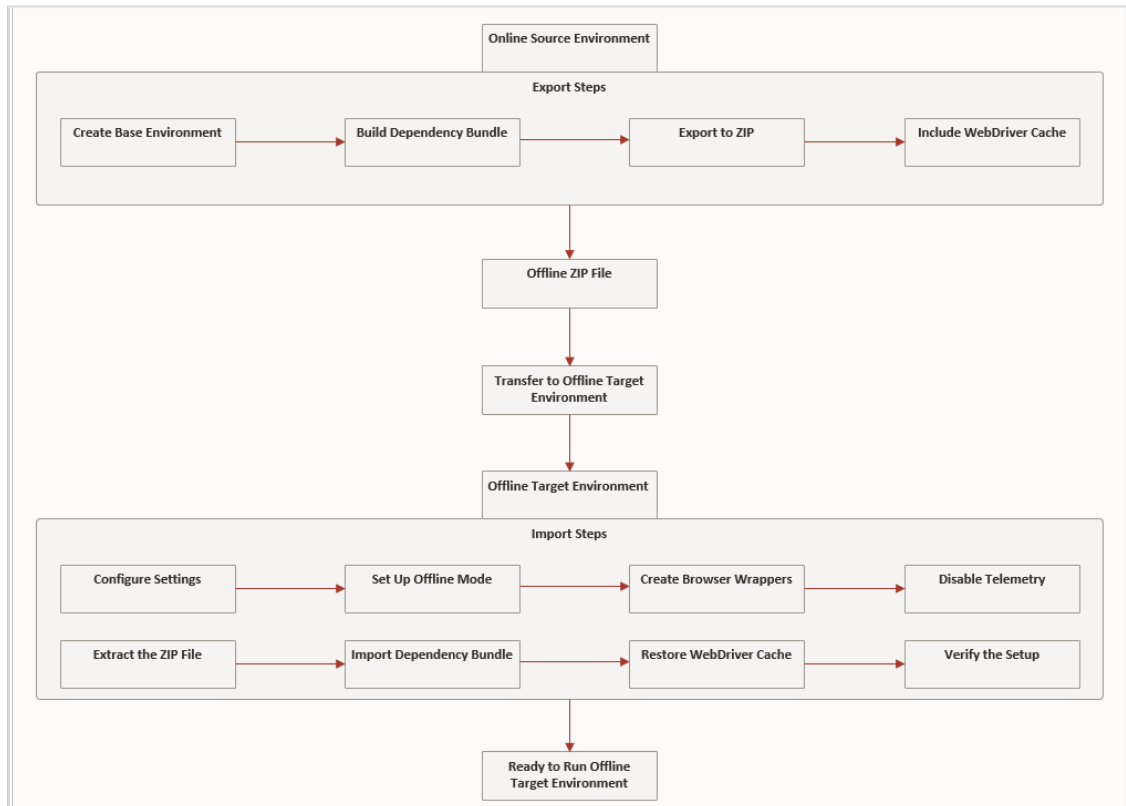
The process ensures that all robots and their dependencies are available in offline mode without internet access by running the scripts which are available in the `scripts` and `offline_scripts` folder of the robot agent ZIP file.

- **Online source environment** is the reference environment with full internet access, where at least one robot has been executed successfully in the desired browser and the export script needs to be executed to generate the exportable ZIP file.
- **Offline target environment** is the air-gapped environment with no internet access or restricted internet access, which is identical to the online source environment in Operating System, CPU architecture, user, other system parameters and where the ZIP file exported from the online source environment should be imported to replicate the robot's virtual execution environment.

Note

Oracle recommends refreshing your offline environments for every release. This can be done by getting the latest robot agent jar file from the online source environment and following the steps outlined in [Export the Robot Agent and Dependencies from the Online Source Environment](#) and [Import the ZIP File to the Offline Target Environment](#).

The following image illustrates the steps executed by the `rpaAgentEnvDiagnosis`, `rcc_offline_export_all`, and `rcc_offline_import_all` scripts described in the following sections.



Important Key Considerations

Consider the following important information before you start the process to execute robot agents in air-gapped environments:

Portability Limitations

The exported ZIP file is self-contained and is ONLY portable across environments with the same Operating System and CPU architecture.

Note

Cross Operating System or cross-architecture imports are NOT supported.

Same User Account Requirement

The SAME user account must be used for:

- Exporting from the online source environment
- Importing into the offline target environment

Note

Using a different user account may cause import failures.

Identical User Home Directory Structure

Ensure that the user home directory paths and structure are IDENTICAL in both the online source and offline target environments.

Note

Any mismatch can lead to runtime errors or import failures.

Browser Version Consistency and WebDriver Handling

The Chrome and, or Firefox browser versions must be identical in both the online source environment and the offline target environment.

Robots are expected to be executed in the online source environment before running the `rcc_offline_export_all` script. During this execution, the latest web drivers corresponding to the installed browsers are resolved. As part of the export and import process, these browser-specific web drivers are copied from the online source environment to the offline target environment. If browser versions differ between environments, the copied web drivers may become incompatible, leading to browser launch or session creation failures.

Export the Robot Agent and Dependencies from the Online Source Environment

Follow these steps to export the robots and their dependencies from the online source environment and create a ZIP file which can be imported into the offline target environment.

1. Prepare the environment.

Ensure that all robots run successfully and all required dependencies are installed.

2. Run the `rcc_agent_env_diagnosis` script to validate the online source environment setup:

- **Linux:** `[AGENT_HOME]/scripts/rcc_agent_env_diagnosis.sh`
- **Windows (PowerShell):** `[AGENT_HOME]\scripts\rcc_agent_env_diagnosis.ps1`

3. Run the `rcc_offline_export_all` script to generate the ZIP file which can be imported into the offline target environment:

- **Linux:** `./rcc_offline_export_all.sh`
- **Windows (PowerShell):** `.\rcc_offline_export_all.ps1`

The script creates a comprehensive ZIP file containing all runtime dependencies required for offline robot execution which you can import into your offline target environment.

Note

- The ZIP file is self-contained and **ONLY** portable across environments with **similar Operating System and CPU architecture**.
- You **MUST** use the same user account for exporting from the online source environment and importing into the offline target environment.

Import the ZIP File to the Offline Target Environment

Follow these steps to import the ZIP file from the online source environment to the offline target environment.

1. Copy the `rpa_agent.zip` file from the online source environment to the offline target environment and unzip the file.
Note: Make sure the user home directory structures are identical in the online source environment and offline target environment.
2. Copy the `<offline-bundle>.zip` file and transfer the ZIP file from the online source environment to the offline target environment.
3. Install the following prerequisites on the offline target environment using temporary internet access or by manually providing the installers:
 - JDK 17
 - Google Chrome
 - (Optional) Mozilla Firefox
4. Run the `rcc_offline_import_all` script to import environments and restore caches:
 - **Linux:** `./rcc_offline_import_all.sh <offline-bundle>.zip`
 - **Windows (PowerShell):** `.\rcc_offline_import_all.ps1 <offline-bundle>.zip`
5. Run the `rcc_agent_env_diagnosis` script to validate the offline target environment setup:
 - **Linux:** `[AGENT_HOME]/scripts/rcc_agent_env_diagnosis.sh`
 - **Windows (PowerShell):** `[AGENT_HOME]\scripts\rcc_agent_env_diagnosis.ps1`
6. Start the robot agent for offline execution: `java -jar agent.jar`.

Once imported, the robot agent can run robots **without any internet connectivity**. The robot agent functions fully offline with robots, runtimes, browser drivers, and all required dependencies available locally.

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

- Accounts for the robot and the robot builder in the application's test environment.
The robot builder should have the same access as the robot.
- Account for the robot in the application's production environment.

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 the 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 applications 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 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 it isn't 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 or its environment.

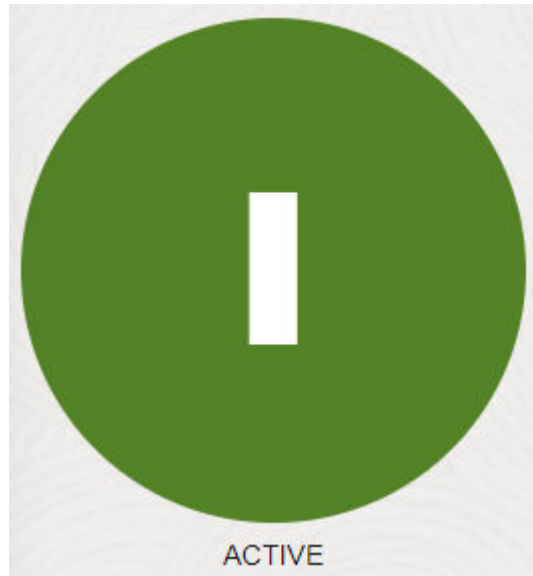
1. Sign in to the Oracle Cloud Console.
2. Navigate to the Integrated applications page.
 - a. Open the navigation menu and select **Identity & Security**. Under **Identity**, select **Domains**.
The Domains page is displayed.
 - b. If it isn't already selected, select the **Compartment** that holds 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 example, 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 Console.
 - b. Open the navigation menu and select **Identity & Security**. Under **Identity**, select **Domains**.

The Domains page is displayed.
 - c. If it isn't 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

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 run on for formal testing.
- The environments that the robots run on in production.

Multiple Robot Agents on an Environment

If you plan to run multiple robot agents on a single VM or computer, download the robot agent only one time to the VM or computer. When you install and start the robot agent, you'll specify the number of robot agents to run. All the robot agents on a single environment run from the downloaded file.

Download the Robot Agent


1. On the computer where you need to install the robot agent, sign in to the Oracle Integration instance that the robot agent needs to connect to.
2. 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.

- Download the robot agent outside a project.
Anyone with access to Oracle Integration can use this option.
 - a. In the navigation pane, click **Design**, then **Agents**.
 - b. Select **Download**, then **Robot Agent**.

- Download the robot agent inside a project.

To use this option, you must have access to a project with an environment pool that has at least one environment associated with it.

- a. In the navigation pane, select **Projects**.
- b. Select the project name.
- c. In the left toolbar, select **Robot** .
- d. In the **Environment Pools** box, point to an environment pool, select **...**, and select **Add environment**.
- e. At the bottom of the Add environment panel, select **Download 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.

3. 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 example, 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.

Note

You can run the `rpaAgentEnvDiagnosis` script to perform a full diagnostic check of your robot environment to check the current environment status. It verifies RCC runtime paths, Chrome and ChromeDriver availability, WebDriver cache, proxy and network connectivity, SSL access, and Chrome launch capability in headless mode.

- Linux: `[AGENT_HOME]/scripts/rpaAgentEnvDiagnosis.sh`
- Windows (PowerShell): `[AGENT_HOME]/scripts/rpaAgentEnvDiagnosis.ps1`

These instructions are the same for running one or multiple robot agents on a virtual machine or computer.

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 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 General Information heading, find the Client ID entry, and copy the value.
Client secret	Below the General Information heading, find the Client secret entry. Select Show secret , and copy the value.
Scope	Below the Token issuance policy heading, and then below the Resources heading, review the entries in the table. If multiple entries appear, copy the value that ends with <code>consumer::all</code> .

2. On the computer where you installed the robot agent, open the folder that contains the agent and its related files.
3. 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=
POLLING=
```

4. Update the properties in the file as needed.

Property	Value to enter
DISPLAY_NAME=	<p>If you're using the default confidential application that Oracle created, this is the only property that you need to define.</p> <p>Enter the name of the robot agent in Oracle Integration.</p> <p>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 the VMs for each robot.</p>
IDCS_CLIENT_ID= IDCS_CLIENT_SECRET= IDCS_SCOPE=	<p>If the robot agent uses a confidential application that your organization created, update the values so that they reference the confidential application:</p> <ul style="list-style-type: none"> IDCS_CLIENT_ID=: Client ID of the confidential application that the robot agent uses. IDCS_CLIENT_SECRET=: Client secret of the confidential application that the robot agent uses. IDCS_SCOPE=: Scope of the confidential application that the robot agent uses. <p>A previous step in this procedure describes how to find these values.</p>
IDCS_URL= SERVICE_INSTANCE_ENDPOINT= SERVICE_INSTANCE_ID=	<p>Do not update the values of any of these properties, even if the robot agent uses a confidential application that your organization created.</p>
POLLING=	<p>Default value is POLLING=fixed. To enable adaptive polling, change the value to POLLING=adaptive.</p> <p>By default, the robot agent uses a fixed polling interval time to check for new messages.</p> <p>You can also use adaptive polling that adjusts the polling frequency based on recent activity. If it is not yet time to poll, the process exits immediately.</p> <p>Whenever new messages are received or an error occurs, the polling interval time is reset to the minimum and held there for a few cycles.</p> <p>If no new messages arrive but the system is still within the time interval window of recent activity, the robot agent continues polling at the minimum interval time. Once activity subsides for an extended period, the robot agent gradually increases the polling interval time using exponential backoff until it reaches a defined maximum time value (30 seconds, by default). After each evaluation, the next poll time is updated based on the current backoff value.</p>

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

Install and Start the Robot Agent

Learn how to install and start the robot agent and then confirm its success.

Install the Robot Agent

After extracting the ZIP file and updating the configuration file, install the robot agent.

1. Build the appropriate command using the following tabs:

- [Windows](#)
- [Mac or Linux](#)

Windows

```
.\scripts\windows\installAgents.ps1 -p agent_install_directory [-n
parallel_agents_count]
```

where:

- *agent_install_directory* is the location of the unzipped files. Oracle recommends not including any spaces in the directory path.
- *parallel_agents_count* is the number of robot agents to run.

Mac or Linux

```
bash scripts/unix/installAgents.sh -p agent_install_directory [-n
parallel_agents_count]
```

where:

- *agent_install_directory* is the location of the unzipped files. Oracle recommends not including any spaces in the directory path.
- *parallel_agents_count* is the number of robot agents to run.

2. Open a command-line tool, and run the command that you built.

You have the following options for command-line tools.

Operating system	Places where you can start the robot agent
Windows	<ul style="list-style-type: none"> • Command Prompt Open a Command Prompt, and navigate to the location of the unzipped files. • Command Prompt as an administrator 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. Run a Command Prompt as an administrator, and navigate to the location of the unzipped files.
MacOS	Open the Terminal app, and navigate to the location of the unzipped files.

Start the Robot Agent

Start the robot agent from the VM or computer on which you extracted the files. A VM or computer can run robots only if a robot agent is running on it.

1. If this is the first time that you're starting the robot agent on the computer, ensure that the computer has access to the public internet.

The robot agent requires this access to download its required dependencies. This requirement exists only for the first time that you start the robot agent.

2. Build the appropriate command using the following tabs:

-
- [Start One Robot Agent](#)
 - [Start Multiple Robot Agents on Windows](#)
 - [Start Multiple Robot Agents on Mac or Linux](#)

Start One Robot Agent

```
.\scripts\windows\startAgent.ps1
```

Start Multiple Robot Agents on Windows

```
.\scripts\windows\startAgents.ps1 -c parallel_agents_count -p  
agent_install_directory
```

where:

- *parallel_agents_count* is the number of robot agents to run.
- *agent_install_directory* is the location of the unzipped files. Oracle recommends not including any spaces in the directory path.

Start Multiple Robot Agents on Mac or Linux

```
bash scripts/unix/startAgents.sh -c parallel_agents_count -p  
agent_install_directory
```

where:

- *parallel_agents_count* is the number of robot agents to run.
- *agent_install_directory* is the location of the unzipped files. Oracle recommends not including any spaces in the directory path.

-
3. Open a command-line tool, and run the command that you built.
 4. Be aware that network issues sometimes prevent the robot agent from starting. If any errors appear in the command-line tool, run the command again, up to a few times if required.

Determine Success

To determine whether the robot agent started successfully, review the information messages in the command-line tool.

- **Review messages in the command line**
 - When the following message appears, the robot agent has started successfully:

INFO - Requesting messages from ControlRoom

- If the following message appears in the command-line tool, the robot agent was already running prior to your attempting to start it:

```
ORPA Agent process is already running.
```

A robot agent might be running already if you configured it to start automatically. See [Start the Robot Agent Automatically](#).

- **Determine the number of running robot agents**

When you install the robot agent, the `agents_pids.properties` is created in the installation directory that you specify in the command.

This file provides the following information:

- The number of robot agents that are running in the directory at any time.
- The start times for the robot agents.
- The process identifiers of the robot agents.

First Time Starting the Robot Agent

After the robot agent starts successfully for the first time, Oracle Integration removes the values of the following properties from the robot agent's configuration file for security properties:

- `IDCS_CLIENT_ID=`
- `IDCS_CLIENT_SECRET=`

Oracle Integration writes the properties to the computer's local keystore and 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.

If the Robot Agent Doesn't Start

If the robot agent doesn't start successfully, troubleshoot the issue.

See [A Robot Agent Doesn't Start](#).

Stop a Robot Agent

As long as its computer is on, the robot agent continues running. You can stop a robot agent by running a command or by shutting down the computer that runs the robot agent.

To stop a robot agent by running a command, open a command-line tool, and run one of the following commands.

Goal	Command
Stop all robot agents When multiple robot agents are running from a directory	<p>Windows:</p> <pre>.\scripts\windows\stopAgents.ps1 -p agent_install_directory</pre> <p>Mac or Linux:</p> <pre>bash scripts/unix/stopAgents.sh -p agent_install_directory</pre> <p>where: <i>agent_install_location</i> is the location of the unzipped files. Oracle recommends not including any spaces in the directory path.</p>
Stop a single robot agent When a single robot agent is running from a directory	<p>Windows:</p> <pre>.\scripts\windows\stopAgent.ps1</pre> <p>Mac or Linux:</p> <pre>bash scripts/unix/stopAgent.sh</pre>

After stopping a robot agent, you can restart it at any time. If you don't want to manually start the robot agent, configure it 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 operating system, configure the robot agent as a service. This task is optional.

Linux Machines

If you're running the robot agent on a Linux machine, simply run the command to start the robot agent. Going forward, the robot agent automatically starts and stops with the operating system.

See [Install and Start the Robot Agent](#).

Windows Machines

Caution

If you don't configure the robot agent as a Windows service, you must manually start the robot agent every time the computer or 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. Update the robot agent's configuration file, and start the robot agent.

The first time you start a robot agent, Oracle Integration removes the values of several properties for security purposes. The values must be in the file when you start the robot agent as a service.

 - a. If the robot agent is running, stop it.
 - b. Open the ZIP file that you downloaded when you downloaded the robot agent.

If you deleted the ZIP file, you can download it again at any time. See [Download the Robot Agent](#).
 - c. In the ZIP file, open the `InstallerProfile.cfg` file in a text editor, such as Notepad.
 - d. Copy the values of the following properties:
 - `IDCS_CLIENT_ID=`
 - `IDCS_CLIENT_SECRET=`
 - e. Navigate to the location where you installed the robot agent, and open the `InstallerProfile.cfg` file.
 - f. Paste in the properties you copied, and remove their empty values.
 - g. Start the robot agent.

See [Install and 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**.

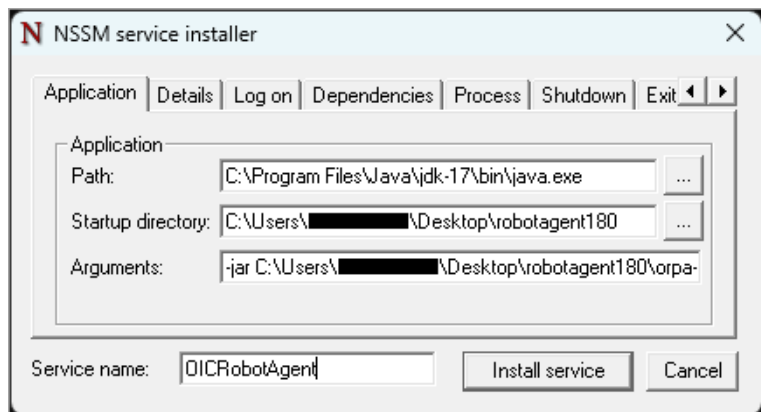
A new row becomes editable in the Edit environment variable dialog.
 - f. Paste the path that you copied in a previous step into the row.

The path shouldn't include `nssm.exe` file name.
 - g. Click **OK**, and then click **OK** again.

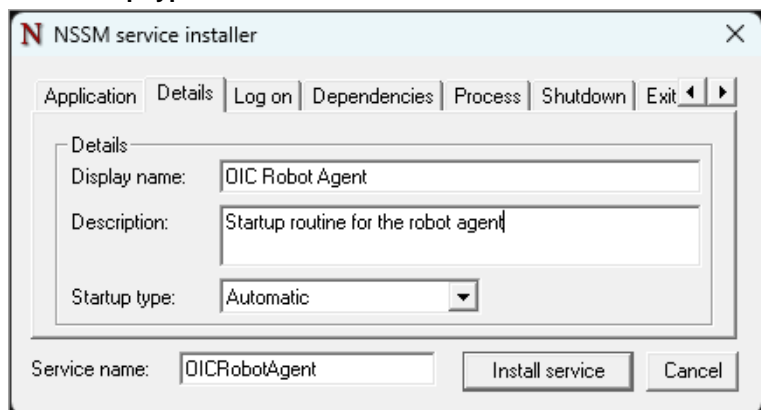
5. Install the robot agent as a Windows service.
 - a. Open a command prompt as an administrator.
 - b. Enter the following command: `nssm install`
The NSSM service installer dialog box appears.
 - c. Fill in the fields.

Tab	Fields to enter
-----	-----------------

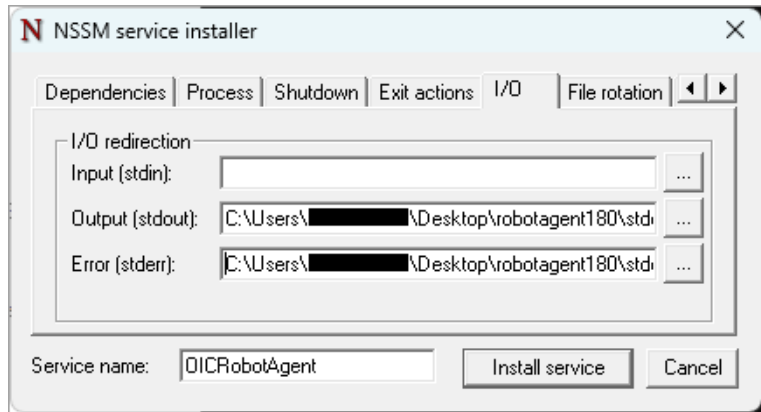
Application	<ul style="list-style-type: none"> • Path: Enter the directory of the <code>java.exe</code> file, including the <code>java.exe</code> file name. • Startup directory: Enter the directory of the robot agent. • Arguments: Enter the command that starts the robot agent, without <code>java</code> at the beginning. If you're not sure, see Install and Start the Robot Agent. • Service name: 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.
-------------	---



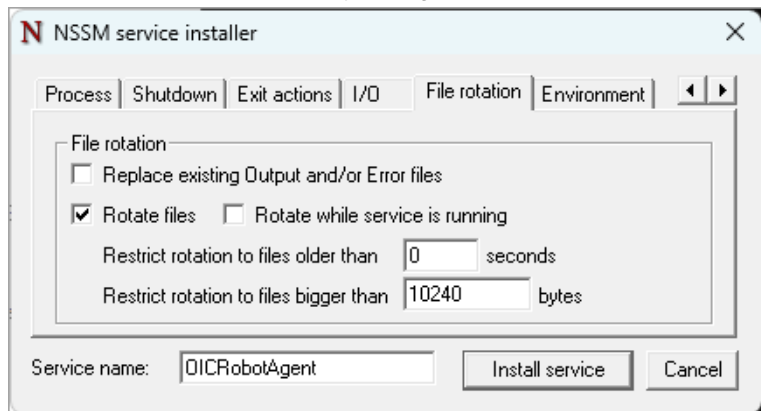
Details	<ul style="list-style-type: none"> • Display name: Enter the display name for the service. • Description: Enter a description for the service. • Startup type: Select Automatic.
---------	---

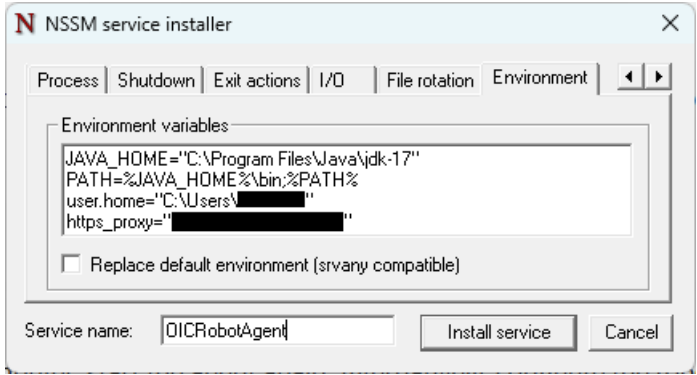


Tab	Fields to enter
I/O	<ul style="list-style-type: none"> • Output (stdout): Select the location for the output files for the service, including the name of the output file. • Error (stderr): Select the location for the error files for the service, including the name of the error file.



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



Tab	Fields to enter
Environment	<p>Environment variables: Enter the following values:</p> <ul style="list-style-type: none"> Location of the Java home, such as: <code>JAVA_HOME="C:\Program Files\Java\jdk-17"</code> <code>PATH=%JAVA_HOME%\bin;%PATH%</code> (Required only if the robot agent is running behind a firewall and is unable to connect to the public internet) Location of the user home, such as: <code>user.home="C:\Program Files\<i><user_name></i>"</code> Definition of the <code>https_proxy</code> parameter. 

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 `services`.

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.

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.

Goal	More information
Learn about your workflow	Workflow for Planning a Robot Workflow for Building a Robot
Watch a video	Video: Introduction to RPA Video: Build a Simple Robot in Minutes Video: See a Simple Robot in Action
Complete a tutorial	Learn with a Tutorial
Read tips and tricks	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. Planning helps ensure that your automation achieves your business goals, and it can reduce or eliminate the need for reworking.

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

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"> Process that isn't working 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. Process that is working 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. 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 example: <ul style="list-style-type: none"> The business process is now automated, with potential for immediate gains in efficiency and effectiveness. You gain insight into how the business process is working, so you can start identifying the areas to improve. <p>Note: 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 example, 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"> If an application doesn't have APIs, you can still automate it using robots. If an application has APIs, you can automate it using integrations, too.
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. Be as specific as you prefer, and work in the format that your team usually uses. For example, 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. Next, determine whether you can deliver all requirements in the initial delivery of the automation. 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 example: <ul style="list-style-type: none"> Identify the number of integrations and robots to design and build. See When to Create Robots Versus Integrations. Specify the actions that the integrations and robots will take. Determine how many environments you must set up, and create system requirements for them. Create a schedule for the work.

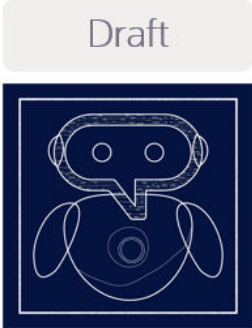
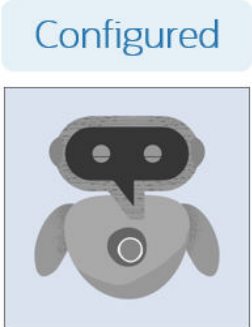
Next workflow: [Workflow for Building a Robot](#).

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	Complete Prerequisites	For example, create accounts, and install and configure the robot agent.
2	Create a Robot Connection Type (Optional)	A robot connection type 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 predefined robot connection types don't meet your needs, create a new one.
3	Create a Robot Connection	Base the robot connection on a robot connection type.

Step	Task	More information
4	Create a Robot	<p>After creating a robot, install the recorder. See Install the Recorder.</p> <p>Next, define the steps that the robot completes by adding robot actions and logic to the robot. See the following:</p> <ul style="list-style-type: none"> • Add an Action to a Robot • Add Logic to a Robot <p>Before or while creating the robot, add the robot resources that the robot uses, such as a trigger or variable. See Create and Update a Robot Resource.</p> <p>While you're building a robot, its status is Draft.</p> <div style="text-align: center;">  </div>
5	Design an Integration That Calls a Robot	<p>A robot developer must create a robot and define a few details, and then an integration developer can start designing an integration. After the robot exists, robot and integration developers can work concurrently or at different times.</p>
6	Fix a Robot's Errors	<p>Address any errors with the robot, such as an incomplete robot action.</p>
7	Specify Where a Robot Runs	<p>Create an environment pool, add computers to it, and associate your robot with the environment pool.</p> <p>If the robot is free of errors, its status changes to Configured.</p> <div style="text-align: center;">  </div>

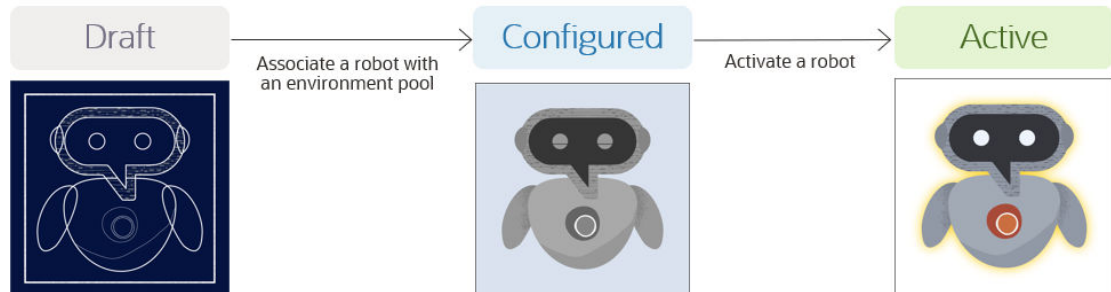
Next workflow: [Workflow for Testing a Robot](#).

Design Time Robot Statuses

A robot's design time status progresses from Draft to Configured to Active. View the status in a project on the Design page.

Workflow

The following diagram illustrates the workflow that you follow to update a robot's status.



Description of the Statuses

Status	Description
Draft	A newly created robot has a status of Draft.
Configured	A robot's status changes to Configured after you complete the following tasks: <ul style="list-style-type: none"> Resolve any errors. See Fix a Robot's Errors. Associate the robot with an environment pool. See Specify Where a Robot Runs.
Activation in progress	After you activate a robot, its status briefly changes to Activation in progress before becoming Active . You typically need to refresh the page to see the Active status.
Active	A robot that is active can be run, either on its own or by an integration. Tip: If you need to update an active robot, just deactivate it. Its status changes from Active to Configured .

Robots *Instances* Have Runtime Statuses

When you run a robot, a robot instance is created, and the robot instance has a status that indicates whether the robot was successful. For details, see [Runtime Statuses of Robot Instances](#).

Video: Build a Simple Robot in Minutes

Watch a video that explores the end-to-end process of building a robot.



[Video](#)

Video: See a Simple Robot in Action

Watch a video that shows how to design the integration that calls a robot, activate and run the integration, watch the robot run, and ensure that everything ran as expected.



[Video](#)

Learn with a Tutorial

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

Tutorial	Description
Get Started with Robotic Process Automation (RPA)	Learn how to build a robot and the integration that calls it.

11 Tips for New Robot Builders

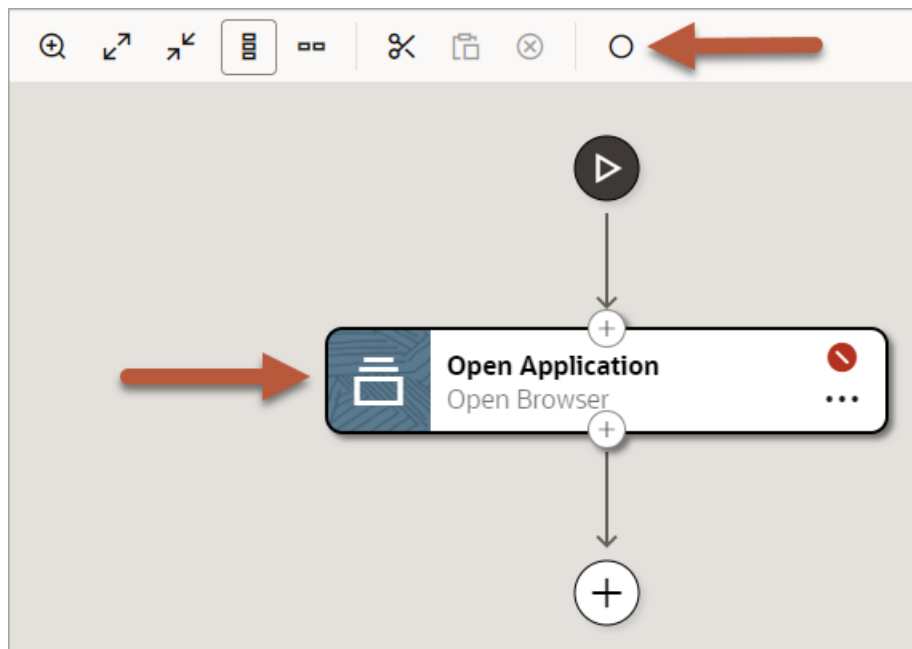
While building a robot, you'll record the actions that a robot completes using the recorder, all while harnessing the power of control and power of the low-code capabilities. Keep reading to learn useful tips and tricks for these building tools.

1. Open the Application to Record in Before Starting 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, just stop and restart the recorder.

2. Select an Action Before Starting the Recorder

To enable the **Record after the selected action** button on the canvas, you must first select an action on the canvas.



3. 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:

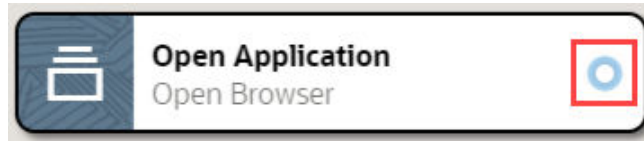
- Foreach loop

When you record from a foreach loop, the recorder asks whether you want to add the action after or within the foreach loop.

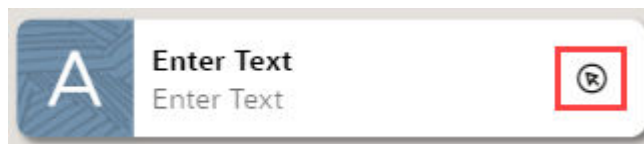
- Switch condition

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.



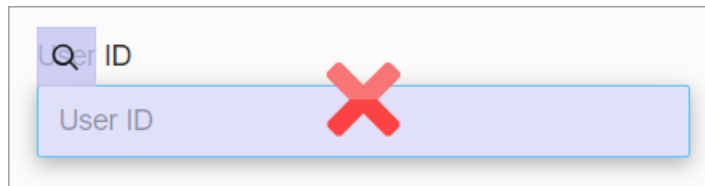
To select a different action, simply point to another action, and select **Place recording cursor**.



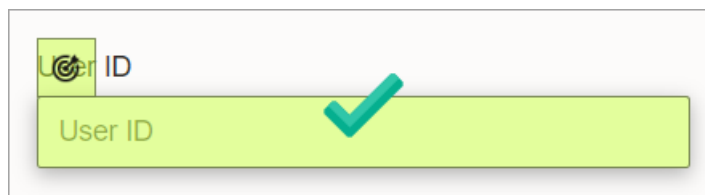
The next action that you record is inserted after the selected action.

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, the entire page becomes shaded green. Cancel the targeting by pressing **Esc**.

6. Target the Correct UI Element

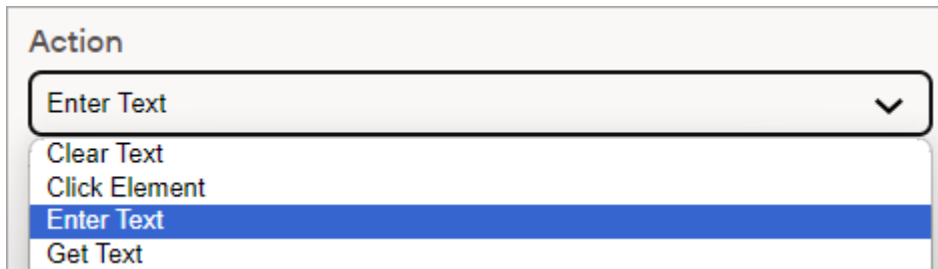
When you point to a UI element while recording, you can often target multiple options, such as the icon on a button, the button itself, and an element that encloses the button. Most of the

time, you can target any of these options, and the robot works as expected. However, if the robot must enter text into a field, select the field, not its label.



7. Select the Correct Robot Action

While you select a field while recording, a drop-down list shows all of the available actions for the UI control.



The recorder selects the action that you are most likely to use, but you can select a different option from the drop-down list.

8. Familiarize Yourself with an Application's HTML

To build a robot using the low-code capabilities, you should have some familiarity with reviewing the HTML of a web page.

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.

This familiarity is helpful because the appearance of a UI element is sometimes different from its underlying HTML code. For example, 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.

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.

For help working with HTML and XPath, see [View HTML and XPath](#).

9. Customize Oracle Integration to Fit Your Working Style

Think about your preferred working style:

- Define the big picture first, and then build the smaller components.
- Define the smaller components first, and use them to construct the big picture.
- Build every component when you need it.

Oracle Integration supports all of these working styles. For example:

- Optimize your working experience to match your preferences by updating your build settings.

See [Tailor Your Building Experience](#).


- Understand your options for building robot resources.

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.

See [Create and Update a Robot Resource](#).

10. To Find Stuff Fast, Use Search


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.


In the toolbar to the right of the canvas, click **Search** .

The Search panel contains a tree that lists all the actions in the robot, including their names and unique identifiers, prefaced with `lc` (`lc1`, `lc2`, and so on).


11. Choose a Layout That Suits You


- If your monitor looks like it belongs in a movie theater, you might prefer to see your robot in a horizontal layout.

On the toolbar above the canvas, select **Horizontal layout** .

Select Vertical layout  to switch back to the original view.

- 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.


On the toolbar above the canvas, select **Maximize** .


Select **Minimize**  to go back to the original view.

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



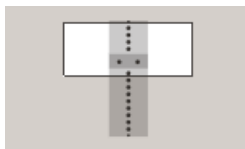
Tailor Your Building Experience

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.

1. Open the robot for which you want to update settings.
 - 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 title bar of the canvas, next to the Save button, select **Action** , and then select **Settings**.

The Settings panel appears.
3. 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"> 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. 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.
Modeling tab	Auto create pre validation page states	<p>When selected, Oracle Integration creates validation that runs before each robot action. The validation is based on the field(s) that you select for the action.</p> <p>When a robot instance runs, the validation ensures that the action starts only after the selected fields are visible in the application.</p> <p>Oracle recommends leaving this setting selected.</p>
Editor tab	Auto edit newly created actions in the canvas	<p>When selected, Oracle Integration opens a panel for specifying details after you add an action or logic to a robot. When not selected, you must double-click an action or logic to view the panel.</p> <p>Whether to leave this setting selected depends upon your workflow. For example:</p> <ul style="list-style-type: none"> If you want to enter all the settings for each action as you build a robot, leave this setting selected. 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.
Editor tab	Auto close palette drawer after drag and drop	<p>When selected, Oracle Integration closes the panel after you add an action to the canvas using the Robot actions button  on the right toolbar, or after you add logic using the Flow control button .</p> <p>Whether to leave this setting selected depends upon your workflow. For example:</p> <ul style="list-style-type: none"> If you want to enter all the settings for each action as you build a robot, deselect this setting. 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>Note: You typically either select or deselect this setting and the previous setting. 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 the canvas. The overview box helps you understand which part of the robot you're looking at.</p>  <p>Select its position from the drop-down.</p>
Editor tab	Show Smart Recording splash screen	<p>When selected, the splash screen appears when you start the recorder for the first time in your sign-in session.</p>

4. Select **OK**.

Create Connections to Applications

To complete its task, a robot must connect to an application. Provide the information that the robot needs, 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](#)

Skip this step if you can use a predefined connection type.

2. [Create a Robot Connection](#)

Create a Robot Connection Type


Oracle Integration comes with a predefined robot connection type that handles 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 type meets 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.

A parameter is data that a robot needs to connect to an application. For example, to connect to a web application, a robot might need the application's URL, credentials, and key. If you're not sure of the fields the application requires, ask the application's administrator.

- 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 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 the robot connection type 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 updates to the values are faster in the future.

Option 1: Work Outside a Robot

You can create a robot connection before you even create a robot.

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](#).

Option 2: Work in a Robot

You can create a robot connection right when you need it, while you're adding an action to a robot. Any action that lets you define a field using a robot connection parameter also lets you create a robot connection.

1. In a robot, add an action that requires a robot connection.
 - a. In the navigation pane, select **Projects**.
 - b. Select the project name.
 - c. In the left toolbar, select **Robot** .
 - d. Create a robot.
See [Create a Robot](#).
 - e. Add an action that requires a robot connection.
For instance, add an open browser action; see [Add an Open Browser Action](#).
 2. In the Open Browser panel, select within the URL field, and select **Robot connections** .
- The Robot connections panel appears.
3. Next to Available robot connections, select **Create** .
 4. Fill in the fields.
 - **Type:** 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](#).

- **Name:** Enter a name for the robot connection. Consider including the name of the application for which you're creating the connection.
 - **URL:** Enter the URL for the web application.
 - **Username:** Enter the user name that the robot uses to sign in to the application.
 - **Password:** Enter the password for signing in.
5. In the Robot connections panel, select **OK**.
- The robot connection is saved. The Robot connections panel refreshes and lists all available robot connections.
6. Continue defining the open browser action.
- See [Add an Open Browser Action](#).

Create a Robot and Integration

You must design an integration to call a robot. Design the integration at any time after creating the robot.

Workflow: Build on Your Own

1. [Create a Robot](#)
2. [Install the Recorder](#)
3. [Design an Integration That Calls a Robot](#)

Workflow: Start from an Accelerator


- [Start from an Accelerator](#)

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](#).

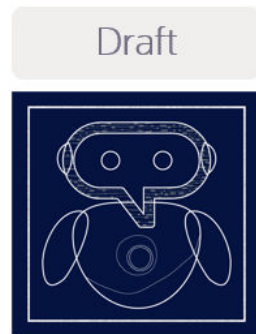
Prerequisite: [Create a Robot Connection](#).

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 browser action that you must define. The new robot has a status of Draft.



4. Define the robot's trigger.

A trigger is a JSON object that defines the interface for the robot. You define an input, which comes into the robot. An input is the activity or event that starts the robot.

You also define an output, which comes out of the robot after it runs. An output is the event or incident that the robot produces.

- 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. Install the recorder.
See [Install the Recorder](#).
7. 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](#)

Install the Recorder

The recorder is an extension that you use to build robots. Everyone who builds robots must install the recorder.

Prerequisite: Create a robot. See [Create a Robot](#).

1. Navigate to the page for installing the recorder.
 - a. Open any robot.
See [Open a Robot](#).
 - b. On the canvas, select the action to record after.
 - c. On the toolbar, select **Record after the selected action** .
 - The Smart record panel appears. If you haven't installed the recorder, **Browser extension not detected** appears.
 - d. Select **Install browser extension**.
2. Select **Add to Chrome**.
3. In the pop-up, select **Add extension**.
The Oracle Robot Flow Designer extension is installed to your browser. A pop-up appears when the installation is finished. Additionally, all browser tabs are automatically refreshed.
4. If you work in an Incognito window, allow the extension in Incognito windows.
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.

Allow in Incognito

Warning: Google Chrome cannot prevent extensions from recording your browsing history. To disable this extension in Incognito mode, unselect this option.



When Oracle uploads an updated version of the recorder to the Chrome Web Store, the Chrome Web Store automatically pushes the update to you. Make sure you install the latest updates as they become available. If you don't, you might not be able to use the latest features.

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.

Prerequisites for Designing an Integration

Before an integration developer can start designing an integration, the robot builder must create a robot and define its trigger, including its input and output properties. These properties represent the contract that the robot makes with the integration. A robot builder can typically complete these tasks in just a few minutes.


See [Create a Robot](#).

Flexible Workflow to Suit Your Schedules

Oracle Integration allows teams to divide their work according to their availability. Additionally, integration and robot developers don't need to worry about block the other person's work:

- An integration developer can start designing the integration that calls the robot after a robot builder creates the robot and defines its trigger. The robot builder doesn't need to specify any actions or logic yet.
- Similarly, a robot developer can build and begin testing a robot, even if the integration doesn't exist yet.

Design an Integration That Calls a Robot







1. Open a project.
 - a. In the navigation pane, select **Projects**.
 - b. Select the project name.
2. In the left toolbar, select **Integration** .
3. 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*.

4. Call the robot from the integration.
 - a. Add a robot action to the integration. You have the following options:
 - Click  at the location where you want to add the action, then select **Robot flow**.
 - On the right side of the canvas, click **Actions**  and drag the **Robot flow** action to the appropriate location.
 - 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.
The drop-down lists all of the robots that are in the same project as the integration.
 - f. Select **Continue**.
 - g. Review the summary, and select **Finish**.
A Map object appears before the Robot process automation action.
5. 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.

Start from an Accelerator

An accelerator is a prebuilt automation solution that can include integrations, robots, and more. You can extend any accelerator by adding a robot or other components to it.

To learn more, see the following resources.

Goal	More information
Find an accelerator	Find Recipes and Accelerators in <i>Using Recipes and Accelerators in Oracle Integration 3</i> .
Install an accelerator	Install a Recipe or Accelerator in <i>Using Recipes and Accelerators in Oracle Integration 3</i> .
Extend an accelerator	Extend an Integration in an Accelerator Project in <i>Using Integrations in Oracle Integration 3</i> .

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 switch to a different window	<p>Close browser: Close an internet browser.</p> <p>Open browser: Open an internet browser and, optionally, sign in to an application.</p> <p>Switch browser: Start working in a different tab or window that is already open.</p> <p>Window: Start working in a window that opens from an application while the robot is running, such as a pop-up window.</p>
Work with an element in a frame	<p>Frame: Interact with a UI element that is in a <code><frame></code> or <code><iframe></code> element in the page's underlying HTML.</p>
Select an element in a user interface	<p>Checkbox: Select or deselect a checkbox, or determine whether a checkbox is selected.</p> <p>Click element: Select a button, link, some items in lists, and some checkboxes.</p> <p>List: Select one or more items in a list or drop-down list.</p>
Interact with text in a user interface, such as obtaining, entering, or clearing text	<p>Alert: Interact with a browser alert dialog box.</p> <p>Clear text: Clear text from a field.</p> <p>Enter text: Enter text into a field.</p> <p>Get text: Get text from the user interface.</p> <p>Radio button: Interact with a radio button.</p> <p>Web table: Get text from one or more columns in a table.</p>
Work with files	<p>Download file: Download a file from an application.</p> <p>Upload file: Upload a file to an application.</p>

Type of interaction	Available actions
Add testing and validation actions	Log : Record an entry in the activity stream. Screenshot : Capture a screenshot and save it as a file. Wait : Pause until a specified period of time has passed. Wait until element is visible : Pause until an element in the user interface is visible.
Manage variables	Data stitch : Manage the variables in a robot.

Get Help with the Settings for Each Robot Action

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

Add an Alert Action

The alert action allows a robot to interact with a browser alert dialog box. For example, the robot can click OK or Cancel in the dialog.

Use Case

A browser alert dialog box prevents you from interacting with a web page. You must interact with the dialog first. The alert action allows a robot to interact with the dialog and optionally capture the message in the dialog.

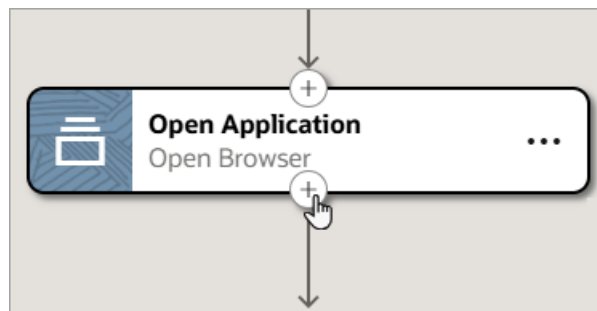
A browser alert dialog box always contains an OK button and sometimes contain a message, Cancel button, and text field.

Add an Alert Action

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 for editing.
See [Open a Robot](#).
2. Add the action to the robot.
 - a. On the canvas, point to an action, and click +.



A menu of available actions appears.

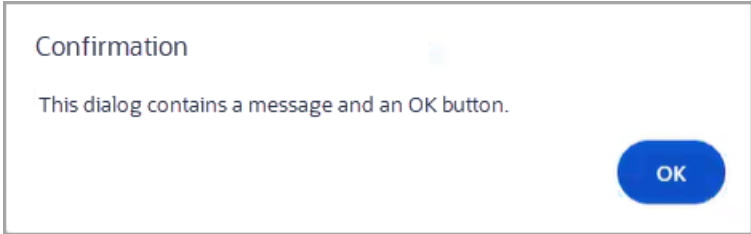
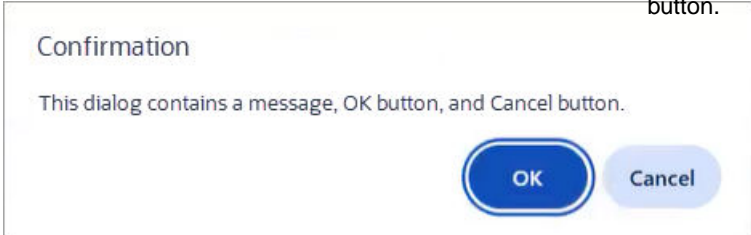
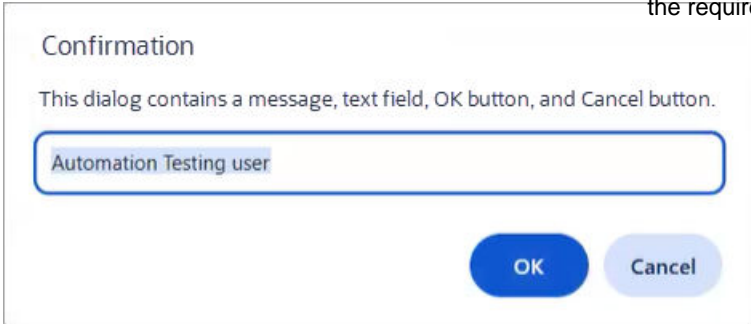
- b. Select **Alert**.

An Alert action appears on the canvas, and the Alert 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**, specify how the robot interacts with the alert.

Operation	Type of browser alert dialog box	Description
Message Alert	This operation interacts with a dialog that contains only one button: OK.	Use this operation to select the OK button.
		
Confirm Alert	This operation interacts with a dialog that contains OK and Cancel buttons.	Use this operation to select the required button.
		
Prompt Alert	This operation interacts with a dialog that contains a text field, OK button, and Cancel button.	Use this operation to enter text and select the required button.
		

Operation	Type of browser alert dialog box	Description
Is Alert Present	This operation interacts with any of the previous dialogs.	<p>Use this operation to check one of the following conditions:</p> <ul style="list-style-type: none"> A browser alert dialog box appears. A browser alert dialog box appears and contains a specific message. <p>If the condition that you check for occurs, the operation returns True. Otherwise, the operation returns False.</p>

- On the **Input** tab, specify how the robot interacts with the dialog box.

The **Input** tab isn't available for the **Message Alert** operation; this operation always selects the **OK** button.

- Text:** (Visible only for the **Prompt Alert** and **Is Alert Present** operations; always optional) Enter an appropriate value:
 - For the **Prompt Alert** operation, specify the text for the robot to enter into the text entry field.
 - For the **Is Alert Present** operation, choose the information that you want to get from the action:
 - To confirm that a specific message appears in the dialog, enter the message that you expect. If the dialog contains this message, the operation returns **True**. Otherwise, it returns **False**.
 - To confirm that a dialog appears, don't enter any text. The operation returns either **True** or **False**, depending on whether a dialog appears.
 - Action:** Specify the button that the robot selects.
 - Accept:** The robot clicks **OK**.
 - Dismiss:** The robot clicks **Cancel**.
 - Leave:** The robot doesn't click any buttons. Use this action with the **Is Alert Present** operation, when you're checking whether a dialog appears or contains specific text.
- (Optional) Save the message from the dialog to a variable.
 - Select the **Output** tab, if it isn't already visible.
 - Click within the **Save to** field, and select **Variables (x)**.
The Variables panel appears.
 - Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#).
 - Select the variable to assign the value to, and drag it to the **Save to** field.
 - Click **OK**.
 - Above the canvas, select **Save**.

Add a Checkbox Action

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

Use Case

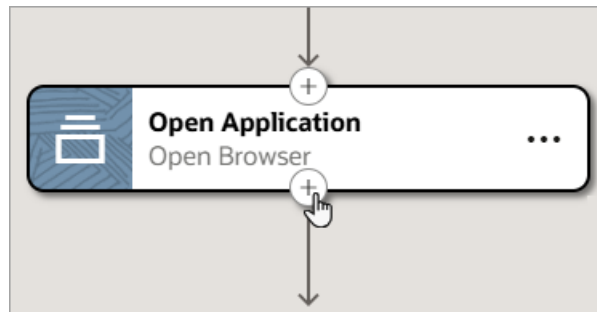
This action has specific requirements for the page's underlying HTML code. You can use the checkbox action only for a UI element with a tag name of `INPUT` and a type attribute of `checkbox`.

Add a Checkbox Action

Note


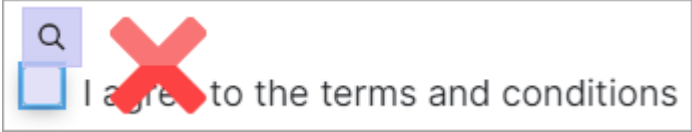
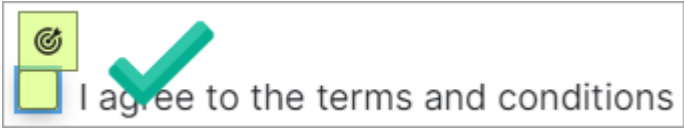

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.


1. Open the robot for editing.
See [Open a Robot](#).
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**, specify 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 determines whether a checkbox is selected. This option is useful if the next action depends on whether a checkbox is selected.
 5. In another browser window or tab, open the application where the robot works.
 6. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. In the application where the robot works, point to the checkbox that the robot needs to interact with, but don't select it yet.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>  <p>d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.</p>  <p>For more tips, see Quick Start for Building Robots.</p> <p>The recorder enters a value in the Locator field in the robot.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
Screenshots	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.
- Assign the value to a variable.
 - a.** Select the **Output** tab.
 - b.** Click within the **Save to** field, and select **Variables (x)**.
The Variables panel appears.
 - c.** Determine whether the variable that you need appears in the list. If not, create it. See [Create a Variable](#).
 - d.** 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.
 - a.** Select the **Output** tab.
 - b.** Click within the **Save to** field, select **More options**, and then select  **Output**.

- The Output panel appears.
- c. Determine whether the output that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
 - d. Select the output 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**.

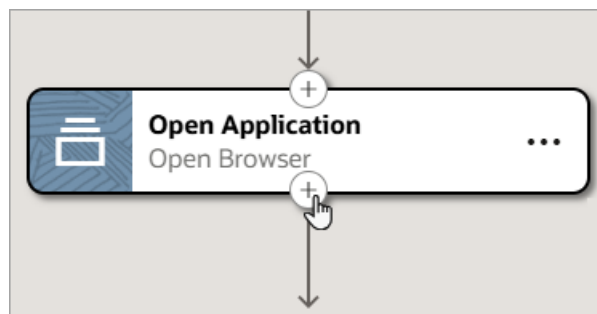
Add a Clear Text Action

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

Note


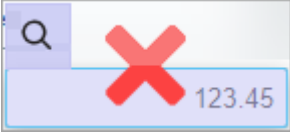
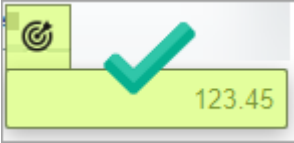

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot for editing.
See [Open a Robot](#).
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. In another browser window or tab, open the application where the robot works.
5. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. In the application where the robot works, point to the field where the clears text, but don't select it yet.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>  <p>d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.</p>  <p>For more tips, see Quick Start for Building Robots.</p> <p>The recorder enters a value in the Locator field in the robot.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
Screenshots	Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots .

- 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](#).

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

Add a Click Element Action

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

Several other actions allow a robot to click and can offer more control for specific interactions. See the [checkbox](#) and [list](#) actions.

Add a Click Element Action Using the Recorder

Before you add the action, consider creating the robot resources that the robot action requires, such as a variable or a trigger's input and output. Otherwise, just pause the recorder and create the resources when you need them.

1. Open the robot for editing.

See [Open a Robot](#).

2. Start the recorder.

- a. On the canvas, select the action to record after.

- b. On the toolbar, select **Record after the selected action** .

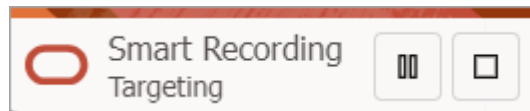
- c. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works. You might need to scroll to find it.

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

The RPA Smart Recording panel appears.

- d. Select **Begin Recording**.

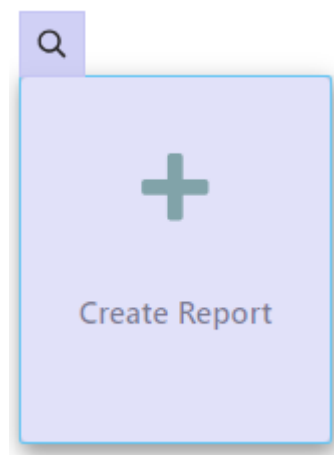
The Smart Recording window appears in the lower-left corner of your browser. Additionally, your mouse cursor can now target elements in the user interface.



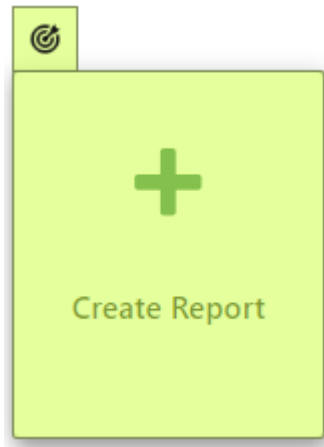
3. Identify the element in the user interface that the robot needs to interact with.

- a. Point to the field that the robot needs to interact with, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- b. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.



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

A panel appears with details about the element you selected.

c. Configure the action.

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

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

- 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.
- 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.

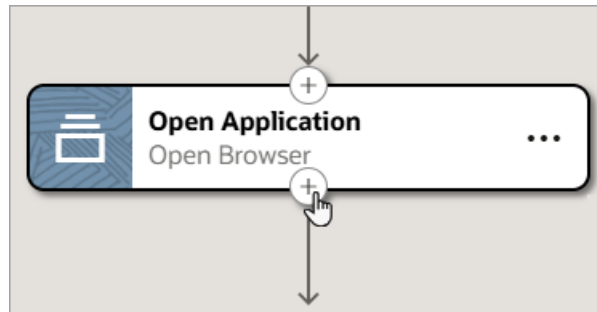
- Customize the action you just added, such as by creating validation or identifying the screenshots to capture. Stop the recording, double-click the action on the canvas, and update the action as needed.

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

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


Add a Click Element Action Using the Low-Code Tools

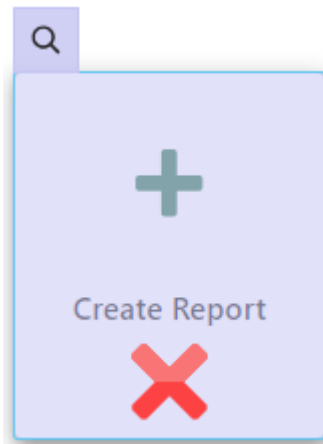
1. Open the robot for editing.
See [Open a Robot](#).
2. Add the action to the robot.
 - a. On the canvas, point to an action, and click **+**.



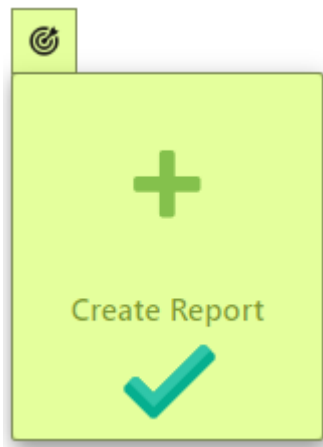
A menu of available actions appears.

- b. Select **Click Element**.
A Click Element action appears on the canvas, and the Click Element 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 another browser window or tab, open the application where the robot works.
 5. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. In the application where the robot works, point to the field that the robot must click, but don't select it yet.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>



- d. Wait for the shading to turn green and the mouse icon to change to a hand, and then 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 these settings](#), if needed. Reusing a target offers benefits.

Field	Description
	<div style="border: 1px solid green; border-radius: 10px; padding: 10px; background-color: #e6f2e6;"> <p>For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
Screenshots	Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots .

- 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](#).

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

Add a Close Browser Action

The close browser action closes the web browser that the robot is working in.

Use Case

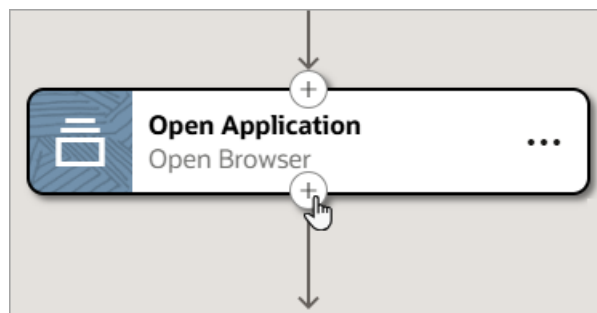
You typically use the close browser action with the switch browser action. For example, get data from one browser, and then switch to a different browser window to paste the values into the application or get more data.

Add a Close Browser Action

i Note

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

- Open the robot for editing.
See [Open a Robot](#).
- Add the action to the robot.
 - On the canvas, point to an action, and click +.



A menu of available actions appears.

- Select **Close Browser**.
A Close Browser action appears on the canvas, and the Close Browser 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.

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.

Use Cases

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. You often use a data stitch when a robot produces an output that is a collection.

A data stitch has two operations: assign and append.

Use case	Description
Assign a value	<p>The assign operation for a data stitch action assigns a new value to a variable.</p> <p>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.</p> <p>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; complete these tasks in a data stitch action.</p>
Append a value	<p>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.</p> <p>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.</p> <p>After the foreach loop processes each invoice, the data stitch action inserts the invoice number into another variable.</p>

For more details and other use cases, see the following:

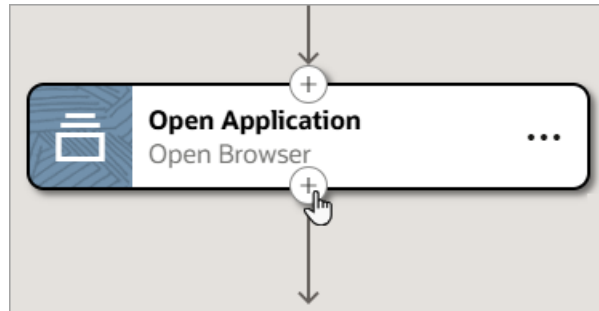
- [Use Case: Save Data After Iterating on Invoices](#)
- [Use Case: Save Values After Iterating on a Table](#)

Add a Data Stitch Action

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 for editing.
See [Open a Robot](#).
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**: The robot must assign a new value to the variable that you selected.
 - **Append**: The robot must preserve the existing value in the variable and append 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.
 6. Click **OK**.
 7. Above the canvas, select **Save**.

Add a Download File Action

The download file action downloads a file so that an integration can act upon the file. The downloaded file is also saved to the home directory of the computer where the robot ran.

Where the File Is Saved

The downloaded file is saved to the following location on the computer or virtual machine where the robot ran:

```
USER_HOME/.orpa/instances/instance_id/output/download
```

Where:

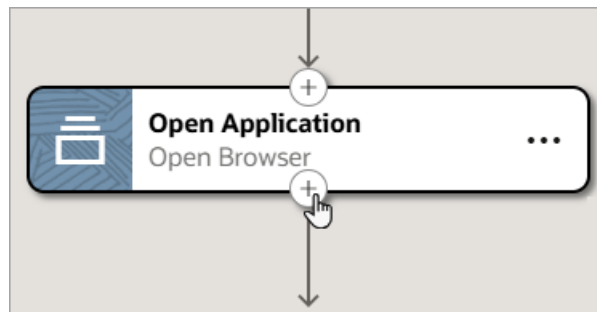
- *USER_HOME* is the home directory, such as `C:\Users\JohnSmith`
- *instance_id* is the identifier of the robot instance

Add a Download File Action

Note


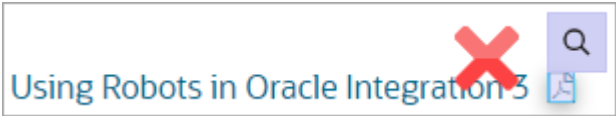
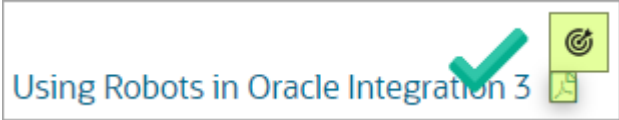

You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.


1. Open the robot for editing.
See [Open a Robot](#).
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 **Download File**.
A Download File action appears on the canvas, and the Download File 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 another browser window or tab, open the application where the robot works.
 5. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. In the application, point to a file link, but don't select it.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>  <p>d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.</p>  <p>For more tips, see Quick Start for Building Robots.</p> <p>The recorder enters a value in the Locator field in the robot.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
Maximum download wait time	<p>Enter the maximum time in seconds to wait for the file to download.</p> <p>If a download attempt isn't successful, the robot retries approximately every 60 seconds, so consider entering a value that is a multiple of 60.</p> <p>The robot moves on to the next action immediately after the download is successful.</p>
Screenshots	<p>Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots.</p>

- 6.** On the **Output** tab, specify the robot resource that you want to assign the file to. You have the following options:
- Assign the file to a variable.
 - a.** Click within the **Save to** 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](#).
 - c.** Select a variable that is of the `File` type and that isn't a collection, and drag it to the **Save to** field.

- Assign the file to an output.
 - a. Click within the **Save to** field, select **More options**, and then select  **Output**.
The Output panel appears.
 - b. Determine whether the output that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
 - c. Select an output that is of the `File` type and that isn't a collection, 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 an Enter Text Action

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

Use Cases

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 any of the following values into the field:

- A variable
- An input parameter
- A hard-coded value

Additionally, the robot can press **Enter** after entering a value.

Add an Enter Text Action Using the Recorder

Before you add the action, consider creating the robot resources that the robot action requires, such as a variable or a trigger's input and output. Otherwise, just pause the recorder and create the resources when you need them.

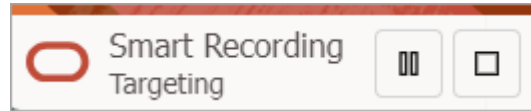
1. Open the robot for editing.
See [Open a Robot](#).
2. Start the recorder.
 - a. On the canvas, select the action to record after.
 - b. On the toolbar, select **Record after the selected action** .
 - c. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works. You might need to scroll to find it.

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

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.



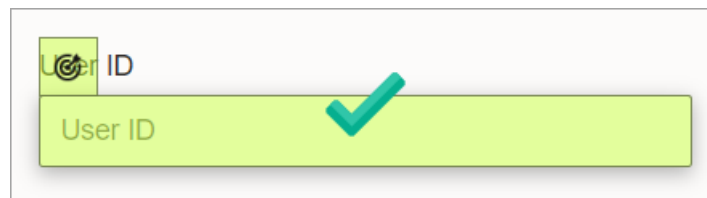
3. Identify the element in the user interface that the robot interacts with.

a. Point to the field that the robot needs to interact with, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



b. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.



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

A panel appears with details about the element you selected.




c. Configure the action.

- **Name:** Enter the name of the target. This text appears on the robot action in the canvas and in the list of targets in the robot.
Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions 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 must 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](#).

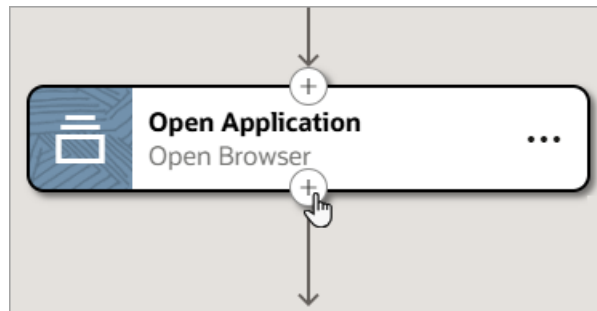
i. Select within the **Value** field, and select **Variables** (x).

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 **Value** field.
- **Test value:** Provide a value to enter into the field now 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](#).
 - 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.
 - 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.
- 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**.



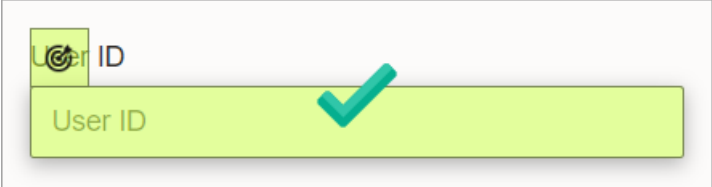

Add an Enter Text Action Using the Low-Code Tools

1. Open the robot for editing.
See [Open a Robot](#).
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 **Enter Text**.
An Enter Text action appears on the canvas, and the Enter 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. In another browser window or tab, open the application where the robot works.
5. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. In the application where the robot works, point to the field where the robot must enter text, but don't click it yet.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>  <p>d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.</p>  <p>For more tips, see Quick Start for Building Robots.</p> <p>The recorder enters a value in the Locator field in the robot.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
Text	<p>Enter the value that the robot inserts into the field.</p> <p>The following instructions are for selecting a variable. To review all of your options for the value, see Define the Fields of an Action.</p> <p>a. Select within the Text field, and select Variables ^(x).</p> <p>The Variables panel appears.</p> <p>b. If the variable to use doesn't exist yet, create it. See Create a Variable.</p> <p>c. From the Available variables list, select a variable, and drag it to the Text field.</p>
Press ENTER key to finish action	<p>If the robot must press Enter after inserting the value into the field, select Press ENTER key to finish action.</p>

Field	Description
Screenshots	Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots .

- 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](#).

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

Add a Frame Action

The frame action lets you interact with any inline frames on a page.

Use the frame action for the following scenarios:

- Interact with inline frames in an `<iframe>` or `<frame>` element in a web page's underlying HTML.

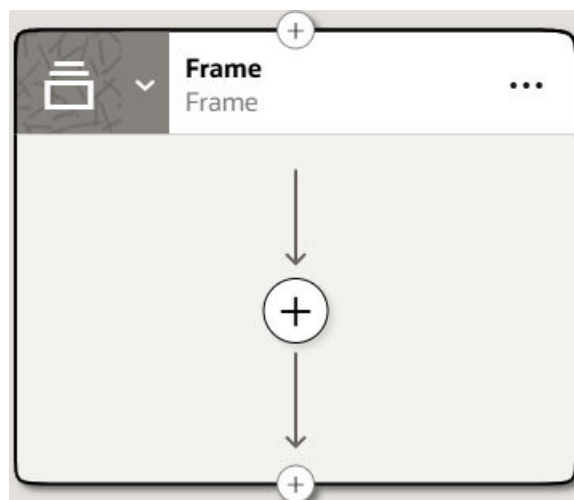
The inline frames can be either direct children of a page or nested inline frames. You typically find `<iframe>` elements on newer websites and `<frame>` elements on older websites.

- Interact with pop-ups for accepting and rejecting website cookies.

Requirements and Restrictions

Familiarize yourself with the frame action's requirements and restrictions before adding it to a robot.

- If you need to interact with UI elements that are in an inline frame, you must use the frame action.
- Add one frame action for each inline frame that a robot interacts with, and include all the actions for the inline frame within the frame container.



- You cannot add the following actions within the frame action:
 - Close Browser
 - Frame

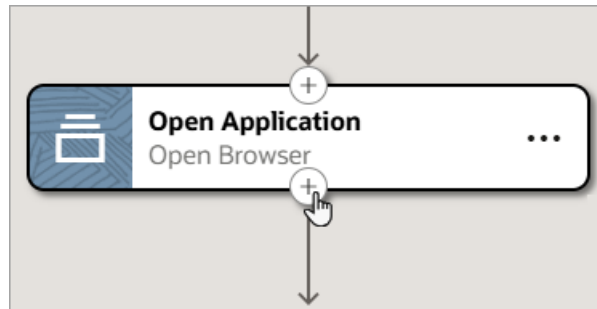
- Open browser
- Switch browser

Add a Frame Action Using the Low-Code Tools

Note


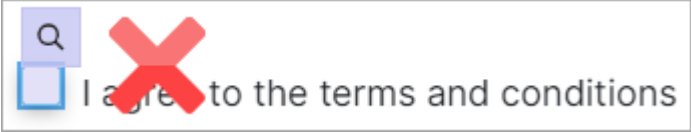
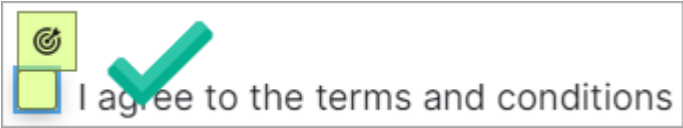


You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot for editing.
See [Open a Robot](#).
2. In another browser window or tab, open the application where the robot works.
3. If the robot needs to work with a UI element that's in a nested inline frame, and the robot must navigate to the nested inline frame, add actions to the robot for the navigation.
For example, if you need to select a tab to show the nested inline frame, add a click element action. See [Add a Click Element Action](#).
4. Add the action to the robot.
 - a. On the canvas, point to an action, and click +.



A menu of available actions appears.

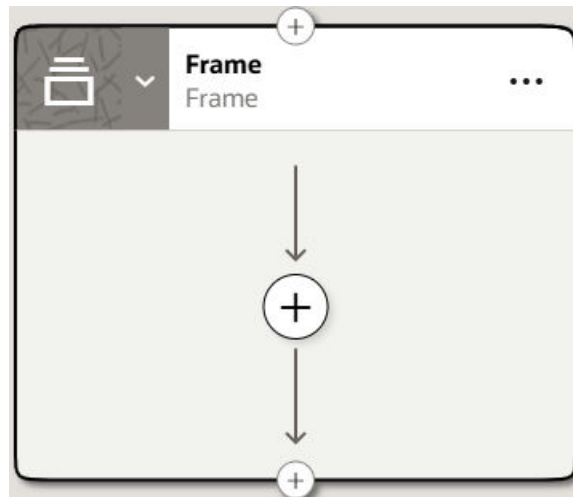
- b. Select **Frame**.
A frame action appears on the canvas, and the Frame panel appears.
5. In the panel, enter a **Name** and **Description** for the action.
The Name appears on the action in the canvas and should help you and others understand the goal of the action.
 6. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. In the application that contains the inline frame, point to any UI element within the inline frame, but don't select it yet.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>  <p>d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.</p>  <p>For more tips, see Quick Start for Building Robots.</p> <p>One of the following events occurs:</p> <ul style="list-style-type: none"> The recorder enters a value in the Locator field in the robot. An error occurs, similar to the following error. In such cases, you must specify the XPath value manually. See An Error Occurs When Targeting an Inline Frame.  <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>

7. Click **OK**.

8. Expand the frame action, and add actions within the frame action.

Include all actions that the robot must perform within the frame. See [Add an Action to a Robot](#).



9. After you finish adding actions, save the robot. 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 verifies that the invoice totals in a report are identical to the actual invoice totals.


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.

For another real-world use case, see [Use Case: Save Values After Iterating on a Table](#).

Add a Get Text Action Using the Recorder

Before you add the action, consider creating the robot resources that the robot action requires, such as a variable or a trigger's input and output. Otherwise, just pause the recorder and create the resources when you need them.

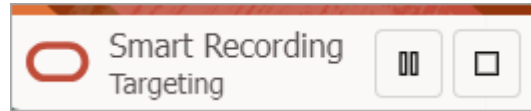
1. Open the robot for editing.
See [Open a Robot](#).
2. Start the recorder.
 - a. On the canvas, select the action to record after.
 - b. On the toolbar, select **Record after the selected action** .
 - c. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works. You might need to scroll to find it.

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

The RPA Smart Recording panel appears.

d. Select **Begin Recording**.

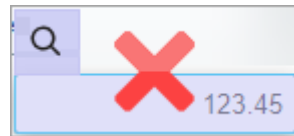
The Smart Recording window appears in the lower-left corner of your browser. Additionally, your mouse cursor can now target elements in the user interface.



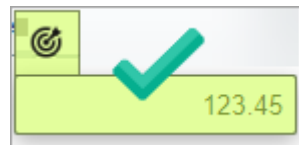
3. Identify the element in the user interface that the robot needs to interact with.

a. Point to the field that the robot needs to interact with, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



b. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.



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





A panel appears with details about the element you selected.

c. Configure the action.

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

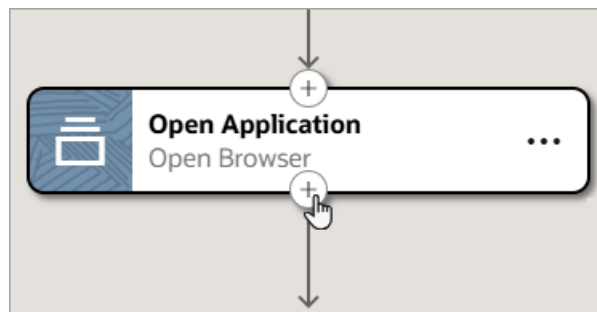
Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions that use the target get the update.

- **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.
 - 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.
 - 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 that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
 - iv. Select the output 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](#).
 - 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.
 - 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.
 - 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 Get Text Action Using the Low-Code Tools

1. Open the robot for editing.
See [Open a Robot](#).
2. Add the action to the robot.
 - a. On the canvas, point to an action, and click **+**.



A menu of available actions appears.


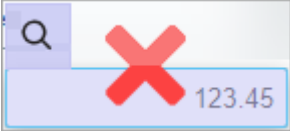
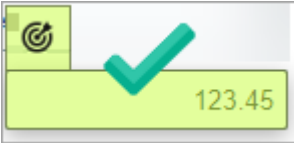

- b. Select **Get Text**.

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.

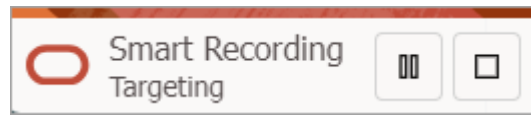
The Name appears on the action in the canvas and should help you and others understand the goal of the action.

4. In another browser window or tab, open the application where the robot works.
5. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<ol style="list-style-type: none"> a. In the robot, click within the Locator field, and select Target a page element . <ul style="list-style-type: none"> The Target a page element panel appears. b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application. <ul style="list-style-type: none"> The application opens. c. In the application where the works, point to the field where the robot gets text, but don't select it yet. <ul style="list-style-type: none"> For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.  <ol style="list-style-type: none"> d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.  <p>For more tips, see Quick Start for Building Robots. The recorder enters a value in the Locator field in the robot.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
Screenshots	Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots .

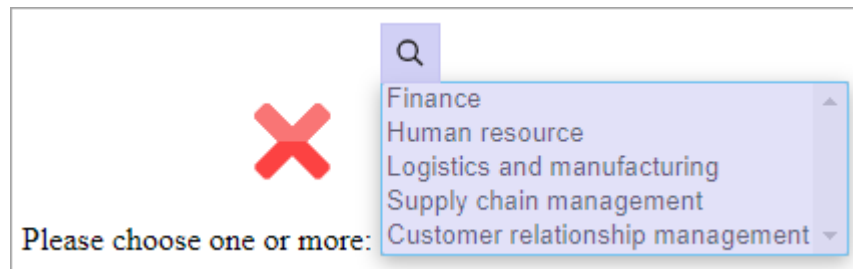
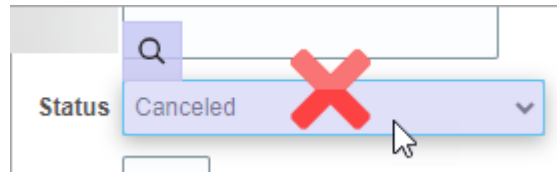
6. On the **Output** tab, specify where to save the text that you get from the field. You have the following options:

- Assign the value to a variable.
 - a. Click within the **Save to** field, and select **Variables** .
 - The Variables panel appears.

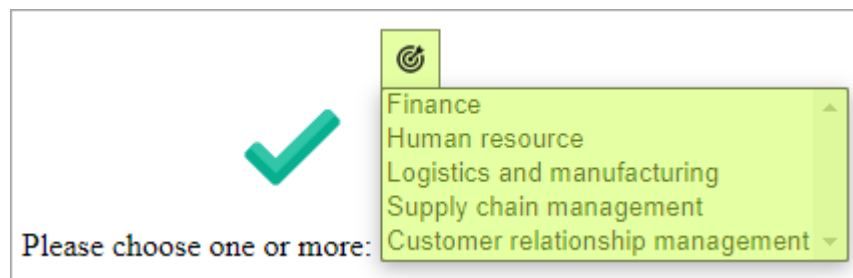
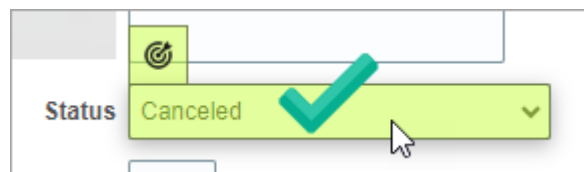


3. Identify the list that the robot interacts with.
 - a. Point to the list, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- b. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.







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

A panel appears with details about the element you selected.

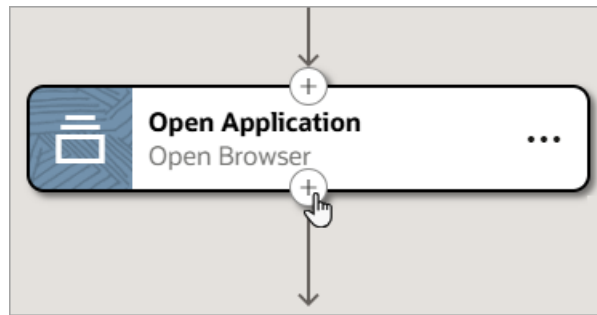
- c. Configure the action.
 - **Name:** Enter the name of the target. This text appears on the robot action in the canvas and in the list of targets in the robot.

Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions 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. See [Operations for the List Action](#).
- d. For **Save to**, if you selected an operation that gets a label or value, specify where to save it. 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.
 - 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.
 - 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 that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
 - iv. Select the output 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](#).
 - 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.
 - 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.
 - 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 for editing.
See [Open a Robot](#).
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.

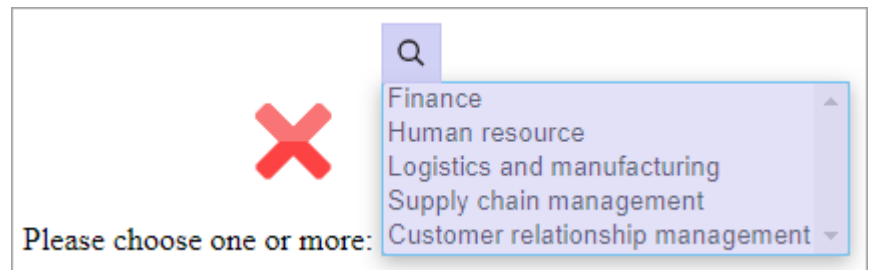
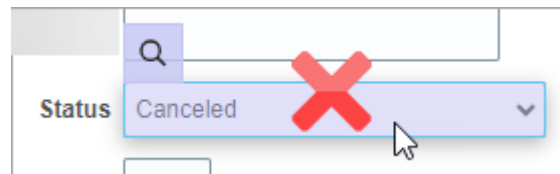
Option	Description
Single Selection List	The list allows you to select only one option. If the robot will select only one option, but the list allows multiple items to be selected, select Multiple Selection List instead.
Multiple Selection List	The list allows you to select one or more options.

5. From Operations, select how the robot interacts with the list. See [Operations for the List Action](#).

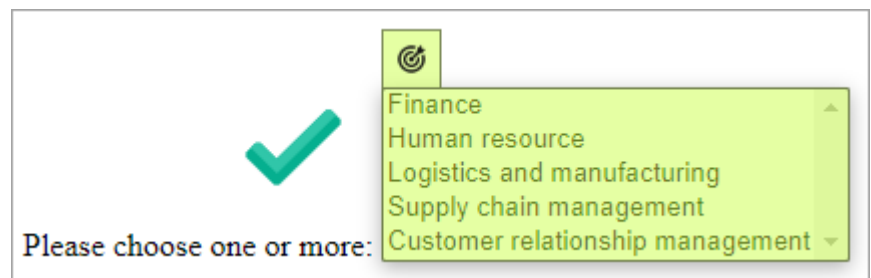
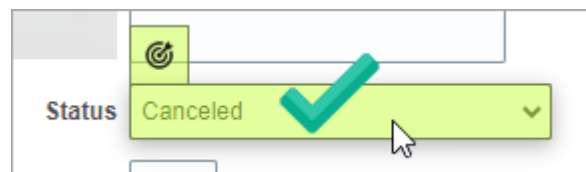
6. In another browser window or tab, open the application where the robot works.

7. On the Input tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. 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.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p> <p>Regardless of the operation that you chose, always point to the list itself.</p>



- d. Wait for the shading to turn green and the mouse icon to change to a hand, and then 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](#)

Field	Description
	<p>these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p>
Screenshots	<p>Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots.</p> <p>If you choose an operation that gets a label or value, Oracle recommends taking a screenshot of the list.</p>

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.
 - a. Click within the **Save to** field, select **More options**, and then select  **Output**.
The Output panel appears.
 - b. Determine whether the output that you need appears in the list. If not, create it. See [Create a Trigger's Input or Output](#).
 - c. Select the output 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**.

Operations for the List Action

Option	Description	Selection list type
Get Selected List Label	Get the label attribute of the selected item. If no items are selected in the list, the robot returns an empty value.	Single Selection List
Get Selected List Labels	Get the label attribute of every selected item. If no items are selected in the list, the robot returns an empty value.	Multiple Selection List
Get Selected List Value	Get the value attribute of the selected item. If no items are selected in the list, the robot returns an empty value.	Single Selection List
Get Selected List Values	Get the value attribute of every selected item. If no items are selected in the list, the robot returns an empty value.	Multiple Selection List
Select From List By Index	Select one or more items in the list. The robot uses the HTML index attribute that you provide to determine the items to select. The indexes of list options start from 0: The first entry in the list is numbered 0, the second entry is numbered 1, and so on.	Single Selection List Multiple Selection List
Select From List By Label	Select one or more items in the list. The robot uses the HTML label attribute that you provide to determine the items to select.	Single Selection List Multiple Selection List
Select From List By Value	Select one or more items in the list. The robot uses the HTML value attribute that you provide to determine the items to select.	Single Selection List Multiple Selection List
Unselect From List By Index	Deselect one or more items in a list. The robot uses the HTML index attribute that you provide to determine the items to select. The indexes of list options start from 0: 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. The robot uses the HTML label attribute that you provide to determine the items to select.	Multiple Selection List
Unselect From List By Value	Deselect one or more items in a list. The robot uses the HTML value attribute that you provide to determine the items to select.	Multiple Selection List

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.

Use Cases

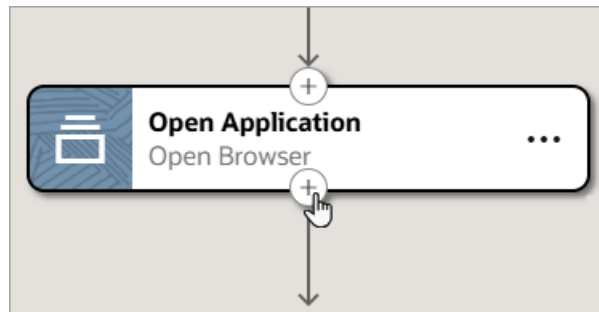
You can record any message to the activity stream, but you typically record a value that the robot obtains. For example, you might record the value from a get text action. The value could be the output of the robot or a value that the robot obtains and later updates using logic.

Add a Log Action

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 for editing.
See [Open a Robot](#).
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 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.

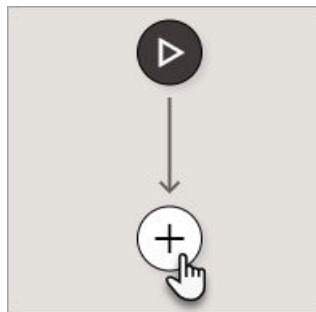
Note

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 for editing.
See [Open a Robot](#).
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.

Field	Description
URL	<p>a. Select within the URL field.</p> <p>This field is for the URL that the robot opens. 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.</p> <p>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.</p> <p>b. In the field, select More options ***, and then select Robot connections .</p> <p>The Robot connections panel appears.</p> <p>c. In the list, expand the entry for a robot connection.</p> <p>d. Drag a parameter for the robot connection to the field.</p>
Browser	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.
Headless	<p>If the browser that the robot runs in should run without launching a visible browser on the computer or virtual machine, select Headless.</p> <p>In a headless browser, the browser page renders as expected, and the robot captures screenshots as expected. However, 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.</p> <p>Additionally, be aware that a robot that runs in a headless browser typically has improved performance.</p> <p>A good practice is to build and test a robot in a non-headless browser, and then use a headless browser in production so that the robot has improved performance.</p>
Full page screenshot after this action	To capture a screenshot after the robot opens the browser, select Full page screenshot after this action .

- 5.** 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.
6. On the **Post Validate** tab, 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**.

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

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


Add a Radio Button Action

The radio button action enables a robot to interact with radio buttons. For example, determine whether *any* radio buttons are selected or a *specific* radio button is selected.

Requirements

You can use this action only when a UI element has the HTML tag `<input type="radio">`.

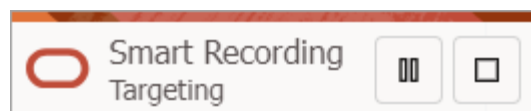
Add a Radio Button Action Using the Recorder

1. Open the robot for editing.
See [Open a Robot](#).
2. Start the recorder.
 - a. On the canvas, select the action to record after.
 - b. On the toolbar, select **Record after the selected action** .
 - c. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works. You might need to scroll to find it.

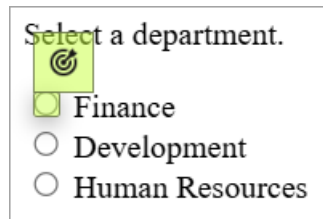
If the application doesn't appear, close the panel, open the application in another tab, and start the recorder again.

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. Point to the radio button. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.



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

A panel appears with details about the element you selected.

4. Configure the action.

a. Fill in the fields.

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


Your [settings](#) determine whether Oracle Integration reuses targets for previously selected UI controls. You can [override these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions 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 radio button. See [Operations for the Radio Button Action](#).

The action that you select determines the other fields that appear.



b. Provide values for the action that you selected.

Action	Fields to update
Is Radio Button Selected	<ul style="list-style-type: none"> • Save to: Choose where to store the result, which is either True or False. You have the following options: <ul style="list-style-type: none"> – Assign the value to a variable. <ol style="list-style-type: none"> Select the Output tab. Click within the Save to field, and select Variables (x). The Variables panel appears. Determine whether the variable that you need appears in the list. If not, create it. See Create a Variable. 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. <ol style="list-style-type: none"> Select the Output tab. Click within the Save to field, select More options, and then select Output. The Output panel appears. Determine whether the output that you need appears in the list. If not, create it. See Create a Trigger's Input or Output. Select the output to assign the value to, and drag it to the Save to field.


Action	Fields to update
Is Radio Button Set To	<ul style="list-style-type: none"> • Value: Specify a radio button. The robot determines whether it's selected. First, choose the radio button's label from the Set Value field, and then select Set Runtime value. • Test value: Optionally, specify a radio button. The recorder determines now whether that specific option is selected right now. First, choose the radio button's label from the Set Value field, and then select Set Runtime value. • Save to: Choose where to store the result, either True or False. You have the following options: <ul style="list-style-type: none"> – Assign the value to a variable. <ol style="list-style-type: none"> i. Select the Output tab. ii. Click within the Save to field, and select Variables (x). The Variables panel appears. iii. Determine whether the variable that you need appears in the list. If not, create it. See Create a Variable. iv. 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. <ol style="list-style-type: none"> i. Select the Output tab. ii. Click within the Save to field, select More options, and then select  Output. The Output panel appears. iii. Determine whether the output that you need appears in the list. If not, create it. See Create a Trigger's Input or Output. iv. Select the output to assign the value to, and drag it to the Save to field.
Select Radio Button	<ul style="list-style-type: none"> • Value: Specify the radio button that must be selected. First, choose the radio button's label from the Set Value field, and then select Set Runtime value. • Test value: Specify the radio button that must be selected now, while you record. First, choose the radio button's label from the Set Value field, and then select Set Runtime value. <p>If needed, you can choose different values for these fields.</p>

c. Select **Save**.

5. Choose the appropriate next step:

- Add another action using the recorder. See [Add an Action to a Robot](#).
- 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.
- 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.

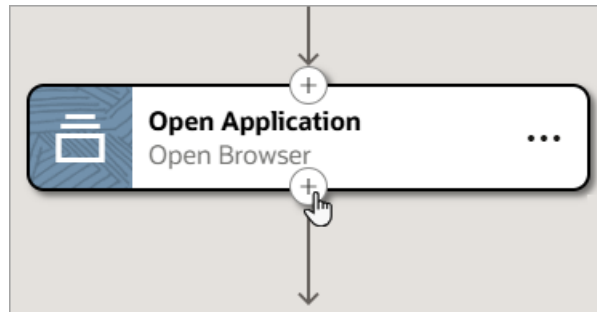
Select **Stop**  in the toolbar of the canvas.

- 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.

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

Add a Radio Button Action Using the Low-Code Tools

1. Open the robot for editing.
See [Open a Robot](#).
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. Configure the action.
 - a. 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.
 - b. From **Operations**, select how the robot interacts with the radio button.
See [Operations for the Radio Button Action](#).
 - c. In another browser window or tab, open the application where the robot works.
4. Specify the information that is passed into the action on the **Input** tab.

Field	Description
Value	<p>If the operation that you selected requires you to target an individual radio button, the Value field appears on the Input tab.</p> <p>If this field appears, specify details for it:</p> <ol style="list-style-type: none"> a. Right-click the radio button that the robot needs to interact with, and select Inspect. b. In the HTML code for the page, locate the <code>id</code> or the <code>value</code> of the radio button. c. Type the value for the <code>id</code> or <code>value</code> into the Value field in the robot. You don't need to format the value in a specific way. For example, if the <code>id</code> is <code>choice1</code>, type <code>choice1</code>.

Field	Description
Group Name	<ol style="list-style-type: none"> a. In the application that the robot needs to work, right-click the radio button group that the robot needs to interact with, and select Inspect. b. In the HTML code for the page, locate the <code>name</code> for the radio buttons in the radio button group. c. Type the value for the <code>name</code> into the Group Name field in the robot.
Screenshots	Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots .

5. If the **Output** tab appears, specify where to save the radio button information from this action. You have the following options:
 - 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. 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 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 to assign the value to, and drag it to the **Save to** field.
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**.

Operations for the Radio Button Action

Option	Description	Example
Is Radio Button Selected	Determine whether <i>any</i> radio buttons in a group are selected.	<p>The action returns True for the following radio buttons:</p> <div style="border: 1px solid gray; padding: 5px; width: fit-content;"> Select a department. <input type="radio"/> Finance <input checked="" type="radio"/> Development <input type="radio"/> Human Resources </div> <p>The action returns False for the following radio buttons:</p> <div style="border: 1px solid gray; padding: 5px; width: fit-content;"> Select a department. <input type="radio"/> Finance <input type="radio"/> Development <input type="radio"/> Human Resources </div>
Is Radio Button Set To	Determine whether a <i>specific</i> radio button is selected.	<p>The action returns True if you're checking whether Development is selected:</p> <div style="border: 1px solid gray; padding: 5px; width: fit-content;"> Select a department. <input type="radio"/> Finance <input checked="" type="radio"/> Development <input type="radio"/> Human Resources </div> <p>The action returns False if you're checking whether Finance is selected:</p> <div style="border: 1px solid gray; padding: 5px; width: fit-content;"> Select a department. <input type="radio"/> Finance <input checked="" type="radio"/> Development <input type="radio"/> Human Resources </div>
Select Radio Button	Select a radio button.	<p>If you tell the robot to select the Development radio button, the robot selects that option:</p> <div style="border: 1px solid gray; padding: 5px; width: fit-content;"> Select a department. <input type="radio"/> Finance <input checked="" type="radio"/> Development <input type="radio"/> Human Resources </div>

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.

Use Cases

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 example:

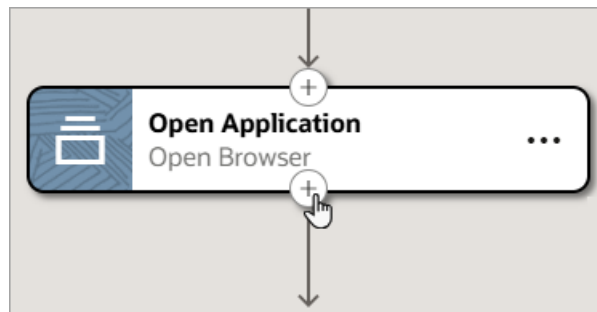
- If an action opens a new tab or browser, use the screenshot action to capture an image of the new tab.
- If you need to send a screenshot to another application, use the screenshot action to capture the image.

Add a Screenshot Action


① 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 for editing.
See [Open a Robot](#).
2. Add the action to the robot.
 - a. On the canvas, point to an action, and click +.

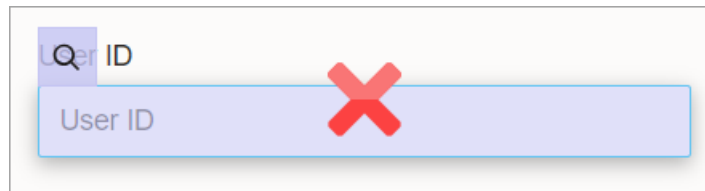


A menu of available actions appears.

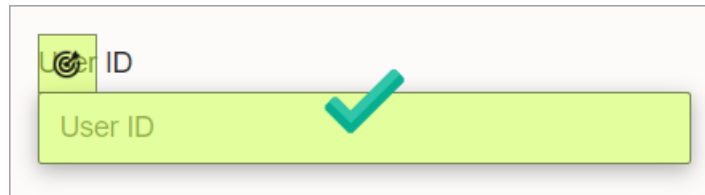
- b. Select **Screenshot**.
A Screenshot action appears on the canvas, and the Screenshot panel appears.
3. In the panel, enter a **Name** and **Description** for the action.
The Name appears on the action in the canvas and should help you and others understand the goal of the action.
 4. Specify how 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 or tab, open the application where the robot works.
 - 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 panel, from the **Select browser tab to target** drop-down, select the application where the robot works, and select **Go**. You might need to scroll down to find the application.
The application 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, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- e. Wait for the shading to turn green and the mouse icon to change to a hand, and then 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 these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions 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 a robot's work in a given window or tab and switches the robot to a different browser window or tab. The window or tab that the robot switches to must be open already.

Use Case

To read about a real-world use case, see [Use Case: Switch Browsers](#).

Alternatives

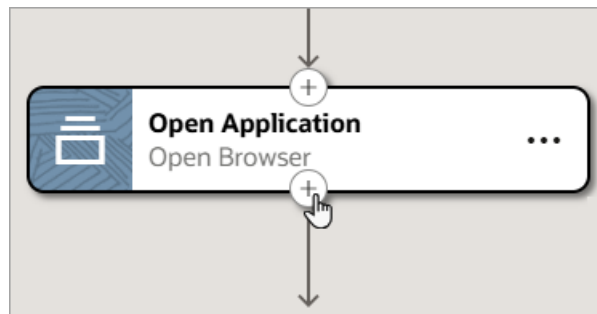
If the robot must interact with a window that opens while the browser is running, such as a pop-up window, use the window action. See [Add a Window Action](#).

Add a Switch Browser Action

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 for editing.
See [Open a Robot](#).
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, fill in the fields.

Field	Description
Browser window	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.
	<div style="border: 1px solid green; border-radius: 10px; padding: 10px; background-color: #e6f2e6;"> <p> Tip</p> <p>To learn more about how to switch between browsers, see Use Case: Switch Browsers.</p> </div>
	<ol style="list-style-type: none"> 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.
Full page screenshot after this action	To capture a screenshot after the robot switches browsers, select Full page screenshot after this action .

5. On the **Post Validate** tabs, specify whether to complete any validation after the action. See [Add Validation to a Robot Action](#).
6. Click **OK**.
7. Above the canvas, select **Save**.

Add an Upload File Action

The upload file action uploads a file to an application.

Prerequisites

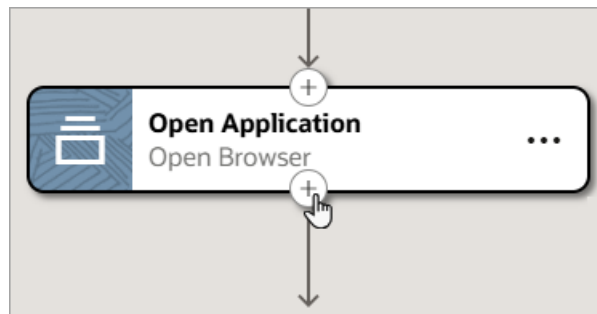
1. Confirm that the web application that you need to upload a file to contains the following buttons:
 - A button for navigating to the file.
 - A separate button for uploading the file.
2. Create a variable that is of the `File` type.
See [Create a Variable](#).
3. Determine where you'll get the file to upload. You have the following options:
 - In the same robot, first download the file from a web application, and then upload the file.
While defining the download file action, specify the variable that you just created as the action's output. See [Add a Download File Action](#).
 - Use an integration to get the file, and then use the integration to pass the file to the robot.
While defining the robot's trigger, create and specify an input property that is of the `File` type. To define the robot's trigger, see [Create a Robot](#).

Add an Upload File Action

Note


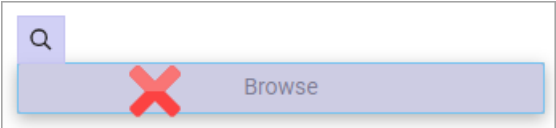
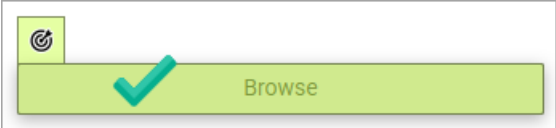


You must use the low-code capabilities to add this action to a robot. Keep reading for step-by-step instructions.

1. Open the robot for editing.
See [Open a Robot](#).
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 **Upload File**.
An Upload File action appears on the canvas, and the Download File 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 another browser window or tab, open the application where the robot works.
 5. On the **Input** tab, specify input details for the action.

Field	Description
browse	<p>a. Click within the browse field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. In the application, point to the button for browsing to a file, but don't select it.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>  <p>d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.</p>  <p>For more tips, see Quick Start for Building Robots.</p> <p>The recorder enters a value in the browse field in the robot.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
File reference	<p>Specify the variable that identifies the file to upload.</p> <p>a. Click within the File reference field, and select Variables .</p> <p>The Variables panel appears.</p> <p>b. Determine whether the variable that you need appears in the list. If not, create it. See Create a Variable.</p> <p>c. Select a variable that is of the File type and that isn't a collection, and drag it to the File reference field field.</p>
submit	Repeat the steps you followed for the browse field, but target the button that you select to upload the file to the web application.
Maximum upload wait time (seconds)	<p>Enter the maximum time in seconds to wait for the file to upload.</p> <p>If an upload attempt isn't successful, the robot retries approximately every 60 seconds, so consider entering a value that is a multiple of 60.</p> <p>The robot moves on to the next action immediately after the upload is successful.</p>
Screenshots	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 Wait Action

The wait action pauses a robot for a specified period of time. Use the wait action sparingly, as it leads to a longer running time for a robot instance.

Use Case

Consider an application that updates its user interface after you select an option from a drop-down. In such cases, you might add a wait action to the robot to ensure that the user interface gets a chance to update before the robot tries to complete its next steps.

Alternatives

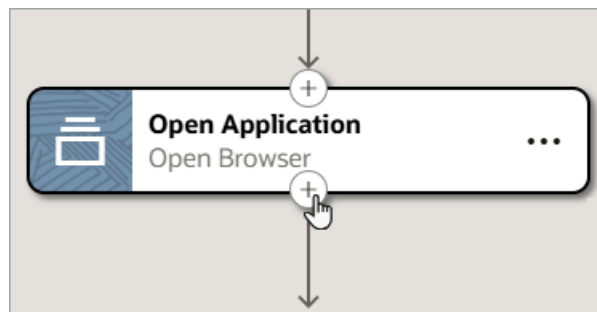
If you want a robot to wait only until a specific element is visible in the user interface, use the "wait until element is visible" action instead. See [Add a Wait Until Element Is Visible Action](#).

Add a Wait Action

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 for editing.
See [Open a Robot](#).
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**.
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.

- On the **Input** tab, fill in the fields.

Field	Description
Timeout	Enter a hard-coded value, such as 3 for a three-second delay. Alternatively, you can specify a variable or input that defines the wait period. See Define the Fields of an Action .
Screenshots	Specify whether to capture any screenshots as part of the action. See Capture Screenshots in Robots .

- 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](#).

- Click **OK**.
- 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.

Use Cases


Every robot action includes optional validation. This validation is more robust than the "wait until element is visible" action and can handle nearly all wait-related activities.

Therefore, most of the time, you do not need to add the "wait until element is visible" action to any robot. For details about the validation on robot actions, see [Add Validation to a Robot Action](#).

If you opt to add this action to a robot, 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

Before you add the action, consider creating the robot resources that the robot action requires, such as a variable or a trigger's input and output. Otherwise, just pause the recorder and create the resources when you need them.

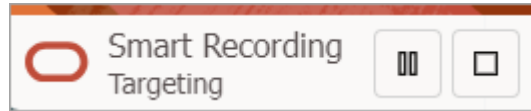
- Open the robot for editing.
See [Open a Robot](#).
- Start the recorder.
 - On the canvas, select the action to record after.
 - On the toolbar, select **Record after the selected action** .
 - In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works. You might need to scroll to find it.

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

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 that the robot needs to interact with, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



b. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.



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

A panel appears with details about the element you selected.

c. Configure the action.

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

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

- 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.
- 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.

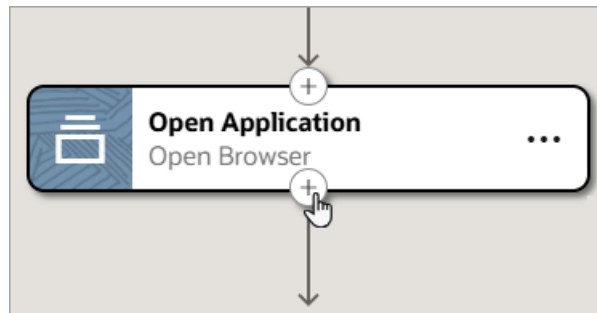
- Customize the action you just added, such as by creating validation or identifying the screenshots to capture. Stop the recording, double-click the action on the canvas, and update the action as needed.

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

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



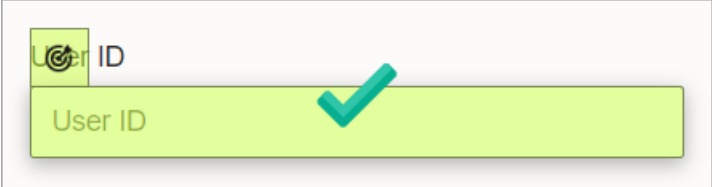

Add a Wait Until Element Is Visible Action Using the Low-Code Tools

1. Open the robot for editing.
See [Open a Robot](#).
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. In another browser window or tab, open the application where the robot works.
 5. On the **Input** tab, specify input details for the action.

Field	Description
Locator	<p>a. In the robot, click within the Locator field, and select Target a page element .</p> <p>The Target a page element panel appears.</p> <p>b. In the panel, from the Select browser tab to target drop-down, select the application where the robot works, and select Go. You might need to scroll down to find the application.</p> <p>The application opens.</p> <p>c. 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.</p> <p>For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.</p>  <p>d. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.</p>  <p>For more tips, see Quick Start for Building Robots.</p> <p>The recorder enters a value in the Locator field in the robot.</p> <div style="border: 1px solid green; padding: 10px; margin-top: 10px;"> <p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p> </div>
Timeout	Enter the maximum amount of time wait in seconds.
Screenshots	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 Web Table Action

The web table action lets you to identify the columns in a table that you are interested in. Add more actions to interact with the table cells.

Use Cases

Consider a table that contains invoice numbers in one column and a hyperlink in another column. Use a web table action to identify the two columns. Then, use a get text action to get each invoice number, and use a click element action to select each link. Include both of these actions within a [foreach loop](#), which lets you iterate on multiple rows of data.

For a real-world scenario, see [Use Case: Save Values After Iterating on a Table](#).

Requirements and Output

- You can use this action only for a UI element that is coded as a `table`.
- You must create a web table definition, but the timing is up to you.

The web table definition identifies the table that the robot interacts with. If the table changes, update the web table definition as needed.

You can create the web table definition either as part of defining the web table action or before.

Then, when you add a web table action to a robot, you choose the web table definition to use.

- This action 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 a column name.
 - The value is the XPath of a row cell.
- Only the [get text](#) and [click element](#) actions can interact with the XPath values that this action returns.

(Optional) Step 1. Create a Web Table Definition

If you want to create the web table definition before adding the web table action, see [Create a Web Table Definition](#).

Otherwise, you can create the web table definition when you add the web table action. Keep reading for step-by-step instructions.

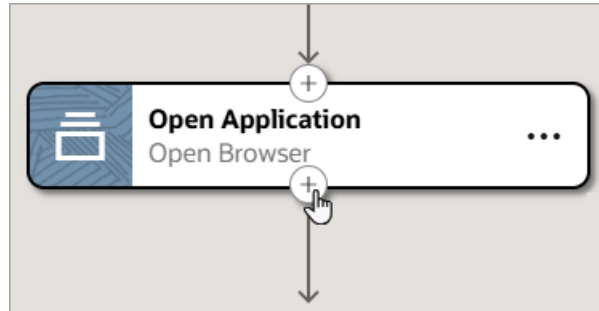
Step 2. Add a Web Table Action

Note

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

1. Open the robot for editing.
See [Open a Robot](#).
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 **Web Table**.

A Web Table action appears on the canvas, and the Web Table 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, perform one of the following tasks.

- If you already created the web table definition that the action interacts with, select it.

- a. Click within the **Web Table** field, and select **Web Table** .

The Web table definitions panel appears.

- b. Drag a web table definition to the **Web Table** field.

- If you haven't created the web table definition yet, create it.

- a. 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 identify it using a web table definition. Instead, use the get text action and save the individual values as a variable collection. See [Add a Get Text Action](#).

- b. Click within the **Web Table** field, and select **Web Table** .

The Web table definitions panel appears.

- c. Select either **Create** (if you have no web table definitions) or **+** (if you have one or more web table definitions).


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

The name should help you and others identify:

- The application where the table appears.
- The page where the table appears.
- If necessary, the table's location on the page.

- e. Identify the first column of data that the robot must interact with.

- f. You don't need to identify every column in the table. Instead, just identify the columns that the robot interacts with.

- i. In another browser window or tab, open the application where the robot works.
- ii. In the web table definition, click within the **Header** field, and select **Target a page element** .

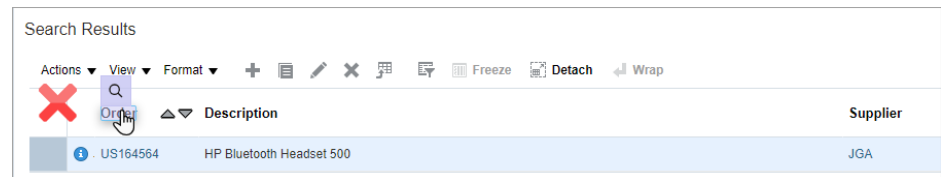
The Target a page element panel appears.

- iii. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works, and select **Go**. You might need to scroll down to find the application.

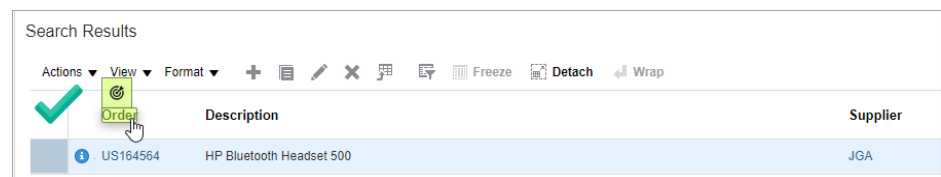
The application opens.

- iv. In the application where the robot works, point to the header of the first column that the robot interacts with, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- v. Wait for the shading to turn green and the mouse icon to change to a hand, and then 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 these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions that use the target get the update.

- vi. Click within the **Row 1** field, and repeat the previous steps to target the first row in the column to target.
- vii. Click within the **Row 2** field, and repeat the previous steps to target the second row in the column to target.
- g. If the robot must interact with one or more additional columns in the table, identify the additional columns.
 - i. Next to Columns, select **Add** .
 - ii. Repeat the previous steps to identify the header and first two rows of the column that the robot interacts with.

- iii. Add more columns as needed until you've identified all the columns that the robot interacts with.
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, review the variable name that the table information is saved to.
The web table action creates a custom data type and variable. The properties of the data type are the columns that you identified in the web table action. The variable is a collection of the data type that the web table action created.
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**.

Your next steps depend upon your goals:

- To iterate over the records in the table, [add a foreach loop](#).
For more guidance, see [Use Case: Save Values After Iterating on a Table](#).
- To record the values from the table to the activity stream, [add a log action](#).
- To view the data type or variable that the web table action created, select **Data types**  or **Variables**  on the toolbar in the canvas.

Add a Window Action

The window action switches a robot's focus to a browser window or tab that opens from the currently selected window, such as a pop-up window.

Use Cases

You can use the window action for the following scenarios.

Scenario	Example
A web application opens multiple child windows	<ul style="list-style-type: none"> • Main window <ul style="list-style-type: none"> – Child window 1 – Child window 2 – Child window 3
A web application opens a child window, which opens a child window, which opens a child window	<ul style="list-style-type: none"> • Main window <ul style="list-style-type: none"> – Child window 1 <ul style="list-style-type: none"> * Child window 2 <ul style="list-style-type: none"> * Child window 3

For a detailed use case that uses this action, see [Use Case: Switch Windows](#).

Alternatives

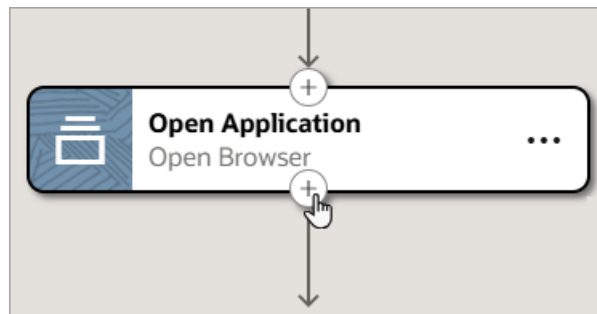
If the robot must interact with a window or tab that is already open, use the switch browser action. See [Add a Switch Browser Action](#).

Add a Window Action

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 for editing.
See [Open a Robot](#).
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 **Window**.
A Window action appears on the canvas, and the Window 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 **Operations** drop-down, select how the robot interacts with a window.

Operation	Description
Close	Close the currently selected window. This operation doesn't switch focus to a different window. If you need to switch focus to another window, finish defining the window action, and then add another window action that uses the Switch To operation.
Get Window Handles	Get an array of strings with the identifiers of all windows that the robot has interacted with up until now. You can store the string values in a collection variable and then use a log action to report the values in the activity stream. See Add a Log Action . Additionally, you can use use this operation to close all open windows. Use a foreach loop to perform this task. See Add a Foreach Loop .

Operation	Description
Get Window Titles	<p>Get an array of strings with the titles of all windows that the robot has interacted with up until now.</p> <p>You can store the string values in a collection variable and then use a log action to report the values in the activity stream. See Add a Log Action.</p> <p>You typically use this operation when troubleshooting a robot that isn't working as expected. The robot might be trying to work in the wrong child window. Getting the titles of all the possible windows can help you fix the robot.</p>
Switch To	<p>Switch to another window or tab that opened from the currently selected window.</p> <p>This operation doesn't close any windows. If you need to close the current window, first close the window using a window action that uses the Close operation. Then, add a window action that uses the Switch To operation.</p>

5. Provide values for the operation that you selected.

Operation	Fields to update
Close	No additional fields.
Get Window Handles	<p>Save to: Select the string collection variable to save the window handles to:</p> <ol style="list-style-type: none"> Select within the Save to field, and select Variables (x). The Variables panel appears. If the variable to use doesn't exist yet, create it. See Create a Variable. You must create a string collection variable. From the Available variables list, select a string collection variable, and drag it to the field.
Get Window Titles	<p>Save to: Select the string collection variable to save the window titles to:</p> <ol style="list-style-type: none"> Select within the Save to field, and select Variables (x). The Variables panel appears. If the variable to use doesn't exist yet, create it. See Create a Variable. You must create a string collection variable. From the Available variables list, select a string collection variable, and drag it to the field.

Operation	Fields to update
Switch To	<p data-bbox="678 212 781 239">Input tab</p> <ul style="list-style-type: none"> <li data-bbox="678 247 1471 449">• Window Id: Choose the window to switch to. If you're switching to a window that opens from the currently selected window, you can usually choose the new window according to its relationship to the currently selected window. However, if you're switching to a previously selected window, or if you're trying to navigate among a few windows, selecting a variable is usually less error prone. <ul style="list-style-type: none"> <li data-bbox="727 464 1398 516">– Choose the new window according to its relationship to the currently selected window. <ul style="list-style-type: none"> <li data-bbox="776 527 1422 617">* NEW: Switch to the window that opened most recently. If the robot is already working in the window that opened most recently, an error occurs when the robot runs. <li data-bbox="776 627 1393 743">* MAIN: Switch to the window that opened the currently selected window. For instance, you might return to the main application window or a window that the main window launched. <li data-bbox="776 753 1459 806">* CURRENT: Don't switch to another window. Instead, remain in the current window. <li data-bbox="727 816 1398 869">– Select the variable that defines the handle of the window to switch to. When you define a Window action that uses the Switch To operation, the Output tab lets you specify a variable that holds the window handle for the currently selected window. For example, when you switch from the main window to its first popup, create a variable named <code>mainWindowHandle</code>, and choose this variable on the Output tab. Whenever you need to return to this window, you can specify this variable as the Window Id for the Switch To operation. <ul style="list-style-type: none"> <li data-bbox="776 1131 1451 1194">a. Select within the Window Id field, and select Variables (x). The Variables panel appears. <li data-bbox="776 1215 1471 1268">b. If the variable to use doesn't exist yet, create it. See Create a Variable. <li data-bbox="776 1289 1471 1341">c. From the Available variables list, select a variable, and drag it to the field. <li data-bbox="678 1352 1471 1457">• Timeout (seconds): Enter the maximum amount of time to wait for the new window to appear. When a robot instance runs, the action completes as soon as the window appears. If the time elapses before the window appears, the robot instance fails. <li data-bbox="678 1478 1471 1520">• Full page screenshot after this action: Select this option if you want to capture a screenshot after the robot opens the window. <p data-bbox="678 1535 1016 1562">Output and Post Validate tabs</p> <ul style="list-style-type: none"> <li data-bbox="678 1570 1471 1932">• Save to: Select the variable in which to store the current window's handle. You can create the variable now. For example, when you switch from the main window to its first popup, create a variable named <code>mainWindowHandle</code>, and choose this variable in the Save to field. Whenever you need to return to this window, you can select this variable as the Window Id for the Switch To operation. <ul style="list-style-type: none"> <li data-bbox="727 1793 1365 1856">a. Select within the Save to field, and select Variables (x). The Variables panel appears. <li data-bbox="727 1877 1471 1932">b. If the string variable to use doesn't exist yet, create it. See Create a Variable.

Operation	Fields to update
	<ul style="list-style-type: none"> c. From the Available variables list, select a string variable, and drag it to the field. • Page state: Specify whether to complete any validation before and after the action. See Add Validation to a Robot Action.

6. Click **OK**.
7. 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"> • Robot connection • Target • Input or output property of a trigger • Variable 	Define the Fields of an Action
Choose the screenshots to capture in a robot action	Capture Screenshots in Robots
Add validation to a robot action	Add Validation to a Robot Action

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.






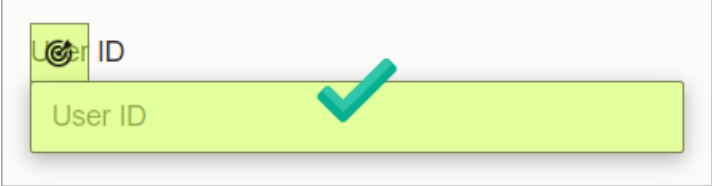

On This Page








- [How to Define Field Values](#)
- [Examples](#)
- [Additional Options When Defining an Action](#)

How to Define Field Values

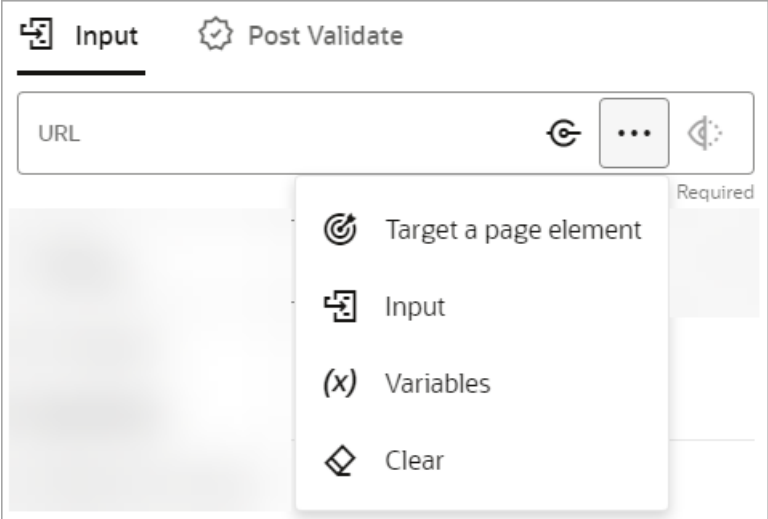
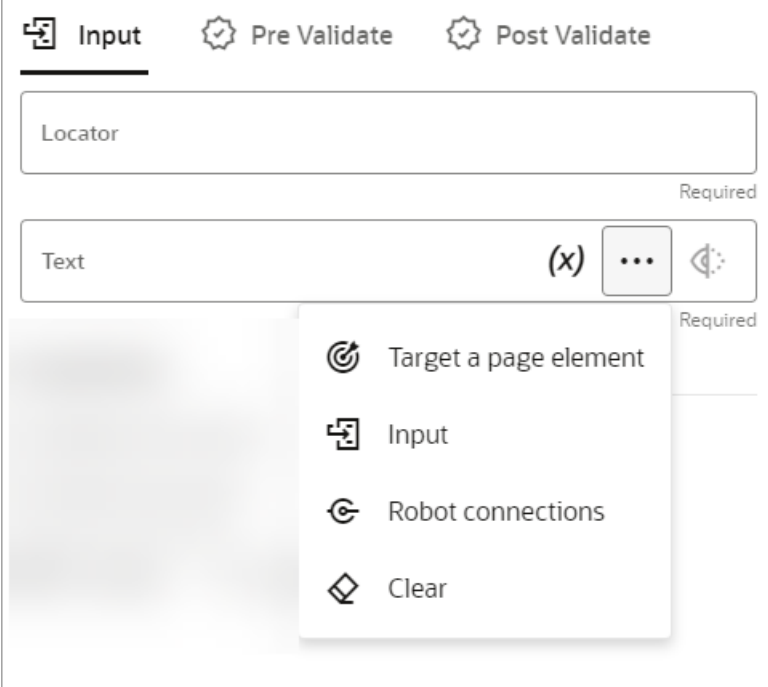
The following table helps you understand all the available options and how to apply them.

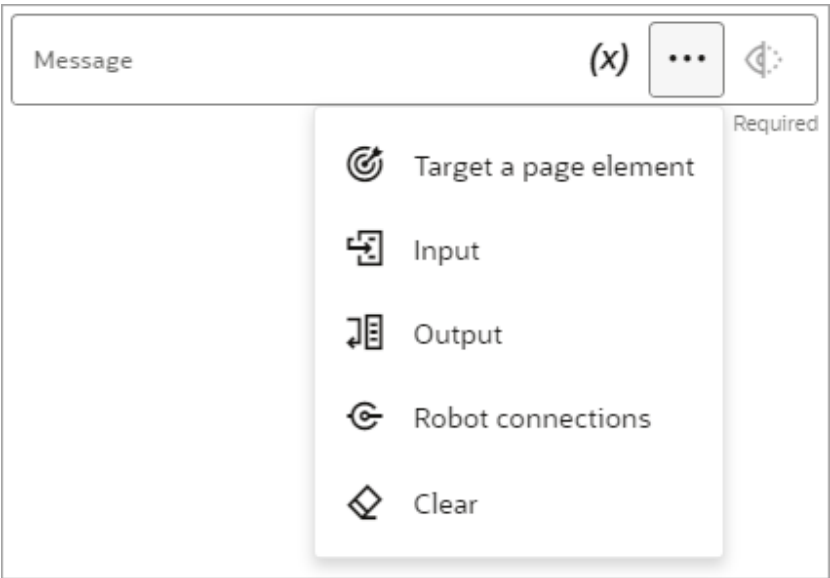
Some fields include only some of these options.

Goal	More information
Use a value from a robot connection  Robot connections	<ol style="list-style-type: none"> In a robot action, select within the field to define, and select Robot connections . This option is sometimes in the More options *** menu. The Robot connections panel appears. In the list, expand the entry for a robot connection. If you haven't created any robot connections, you can create one now. See Create a Robot Connection. Drag a parameter for the robot connection to the field.
Choose an element in a user interface  Target a page element	<p>To select a field using the recorder:</p> <ol style="list-style-type: none"> In a robot action, select within the field to define, and select Target a page element . This option is sometimes in the More options *** menu. The Target a page element panel appears. In the Select browser tab to target drop-down, select an application. In the application, point to the UI element that the robot needs to interact with, but don't select the field yet. For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.
	
	<ol style="list-style-type: none"> Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.
	
	<p> Tip</p> <p>Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings, if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.</p>

Goal	More information
Use an input property from the robot trigger  Input	<ol style="list-style-type: none"> 1. In a robot action, select within the field to define, and select More options , and then select Input . The Input panel appears. 2. From the Properties list, select a property, and drag it to the field. If you haven't created the input yet, you can create it now. See Create a Trigger's Input or Output.
Use an output property from the robot trigger  Output	<ol style="list-style-type: none"> 1. In a robot action, select within the field to define, and select More options , and then select Output . The Output panel appears. 2. From the Properties list, select a property, and drag it to the field. If you haven't created the output yet, you can create it now. See Create a Trigger's Input or Output.
Use a project-level variable (x) Variables	<ol style="list-style-type: none"> 1. In a robot action, select within the field to define, and select Variables (x). This option is sometimes in the More options  menu. The Variables panel appears. 2. If the variable to use doesn't exist yet, create it. See Create a Variable. 3. From the Available variables list, select a variable, and drag it to the field.
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"> 1. In a robot action, select within the field to define. 2. Enter the value for the field.
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 Guidelines for Expressions.</p>

Examples

Action	Example
Open browser	<p>The URL field has offers several options for defining a value.</p> 
Enter text	<p>The Text field offers several options for defining a value.</p> 

Action	Example
Log	The Message field offers several options for the data to include in the activity stream.
	

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 typically most helpful when you are debugging and troubleshooting a new robot that is failing.

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.

Keep in mind that a robot that captures screenshots sometimes runs a little slower and captures more data than a robot that doesn't capture screenshots.

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

Option	Description	When the screenshot appears in the activity stream
Before this action	Take a screenshot of the application before the action occurs.	To include the screenshot in the activity stream, you must also select Log screenshots to activity stream when activating the robot. See Log Screenshots to Activity Stream .
After this action	Take a screenshot of the application after the action occurs.	To include the screenshot in the activity stream, you must also select Log screenshots to activity stream when activating the robot. See Log Screenshots to Activity Stream .
Specify screenshot type	Select the scope of the screenshot: <ul style="list-style-type: none"> • Full page: Capture the entire application screen. • Action target: Capture only the UI element that the action affects. 	N/A

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](#).

If you add this action to a robot, a screenshot always appears in the activity stream, regardless of whether you select **Log screenshots to activity stream** when you activate a robot.

Add Validation to a Robot Action

When you target a UI element in a robot action, Oracle Integration can automatically create prevalidation for it.

Supported Validation

You can add validation that waits to perform the action until the following activities have occurred:

- A UI control becomes visible.
- A UI control becomes enabled.
- A specific amount of time has passed.

About Automatic Prevalidation

Oracle Integration automatically creates prevalidation when you select the **Auto create pre validation page states** setting. See [Tailor Your Building Experience](#).

Here's what's included and not included in the automatic validation:

- **Included:** Confirm that the UI control that you targeted is *visible* before proceeding with the action.

For example, you targeted a Submit button, the validation allows the action to start only after the Submit button is visible.

Because the recorder and low-code tools capture three selectors for each element that you target, the prevalidation condition includes three selectors, one for each element.

- **Not included:** Wait until a specific button is enabled, or wait for a specific period of time before proceeding with the next action.

You can add these checks yourself by following the steps on this page.

All the validation conditions must be met for an action to proceed.

Add Validation to a Robot Action

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 add validation before the robot performs an action, select the **Pre Validate** tab.
 - To add validation after the robot performs an action, select the **Post Validate** tab.
4. If you already created a page state, select it from the **Page state** drop-down.
If you don't need to add any more validation, you're now finished.
5. If you haven't create a page state already, create it now.

A page state is a container for one or more validation checks. After creating a page state, 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 Submit button enabled .
Message	Enter the message that appears when the validation condition isn't met before the timeout period elapses. For example, The Submit button did not become enabled within the specified 30-second timeout period . If the condition times out, the robot fails, and this message appears in the activity stream.

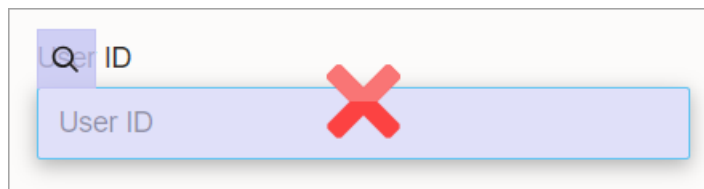
Field	Description
Strategy	<p>Select one of the following options:</p> <ul style="list-style-type: none"> Element: Add validation that is related to the visibility or enabled state of a UI element. Fixed Time: Add a predetermined wait period. <p>Caution: The Fixed Time validation introduces explicit slowness. Consider your requirements carefully before adding this validation, and use it sparingly.</p>
State	<p>(Visible only if you select Element for the Strategy)</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> Visible: The validation condition checks whether a UI element appears before the action proceeds. Enabled: The validation condition checks whether a UI element is clickable before the action proceeds.
Timeout (seconds)	<p>Enter the timeout period for the validation.</p> <ul style="list-style-type: none"> If you selected Element for the Strategy: 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. If you selected Fixed Time for the Strategy: Enter the specific wait time that the robot waits every time it runs.

- e. If you selected **Element** for the **Strategy**): Next to Selectors, click **Create +**, and identify the UI control for the validation.

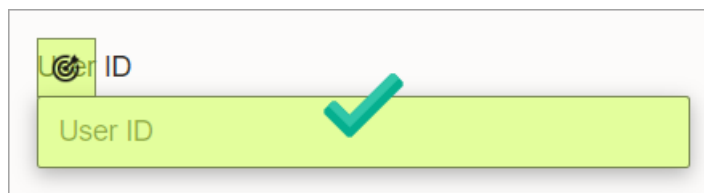
For example, if the validation ensures that a button is visible, target the button.

- i. In another browser window or tab, open the application where the robot works.
- 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, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- iv. Wait for the shading to turn green and the mouse icon to change to a hand, and then 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 these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions 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	Add a Foreach Loop
Take different actions based on conditions	Add a Switch Condition
Stop running	Add a Stop

Add a Foreach Loop

When a robot performs the same actions for multiple items, add the actions to 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 Cases

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.

For more details and for other use cases that use a foreach loop, see the following:

- [Use Case: Update a Set of Invoices](#)
- [Use Case: Save Data After Iterating on Invoices](#)
- [Use Case: Save Values After Iterating on a Table](#)

Prerequisites

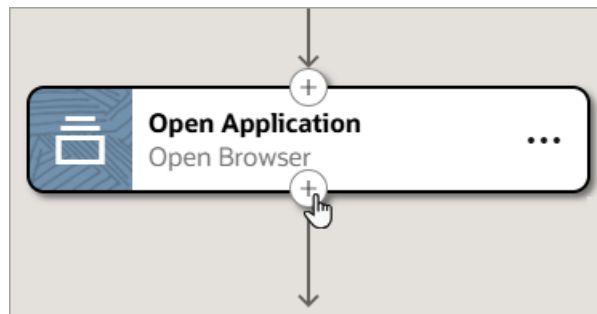
A foreach loop requires a collection variable, which is an array of values. For example, a variable can contain the invoice numbers that require updates. Define the variable at any time, including while you're adding the foreach loop.

Add a Foreach Loop

Note

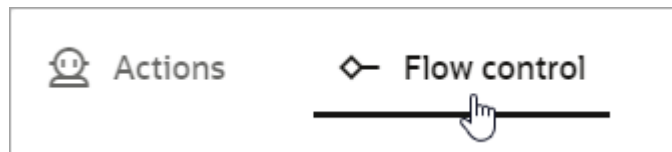
You must use the low-code capabilities to add this logic to a robot. Keep reading for step-by-step instructions.


1. Open the robot for editing.
See [Open a Robot](#).
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** (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](#). 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 Input 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 example, 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. Click **OK**.
8. 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.

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.

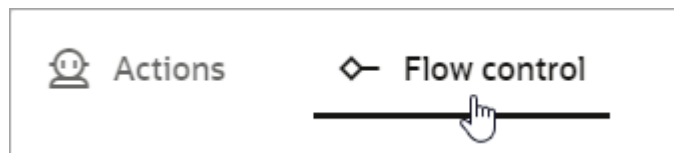
For more details about this use case, see [Use Case: Update a Set of Invoices](#).

Add a Switch Condition

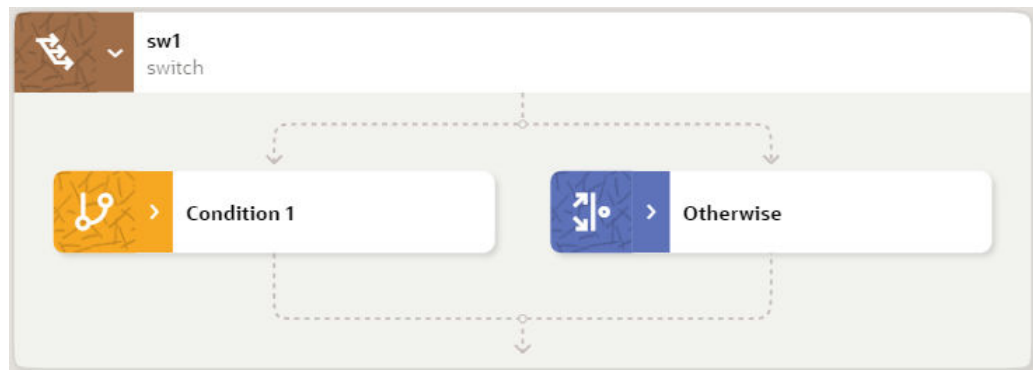
Note

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 for editing.
See [Open a Robot](#).
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.

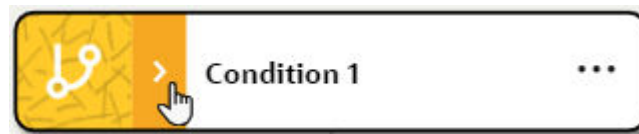
For example, the condition might include an expression to determine whether an invoice meets specific criteria.

- a. Within the switch condition 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:

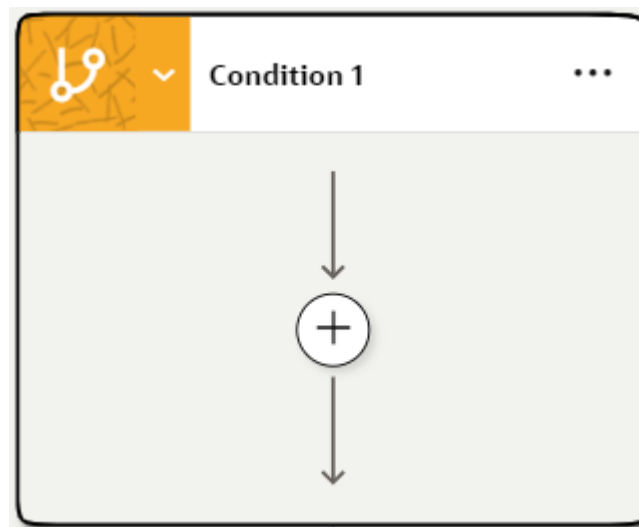
- **Name:** Enter a name for the condition. The name appears in the box for the condition on the canvas.
- **Description:** Enter a description for the condition.
- **Condition:** Enter the expression that the condition must evaluate. If the expression returns `true`, the switch performs the actions 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 [Guidelines for Expressions](#).

- Click **OK**.
- Add one or more actions to the first condition.
 - Point to the condition, and select the > button to expand it.




The condition expands, and a plus sign appears in it.





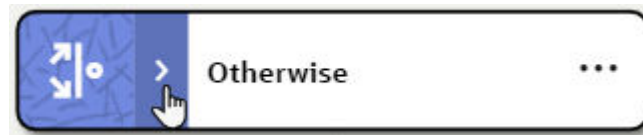
- Define actions or logic that the robot takes if the condition returns `true`.
See [Add an Action to a Robot](#) and [Add Logic to a Robot](#).
- If needed, add more conditions to the switch.

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.

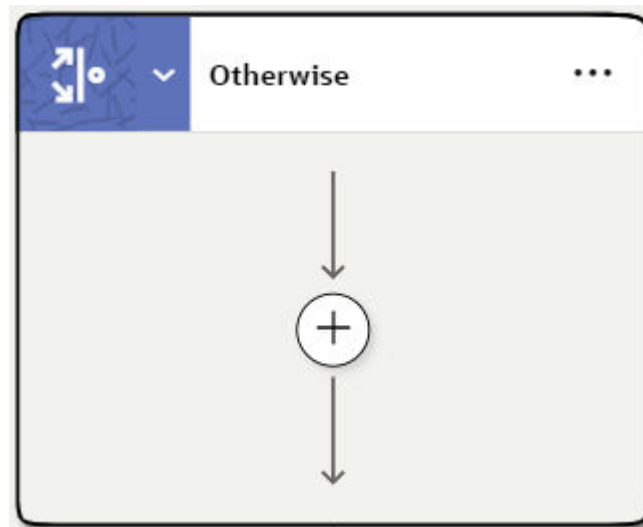
 - Point to the switch condition on the canvas, and select **Insert switch condition** .
 - Use the previous steps to define the condition and its actions and logic.
 - Define the otherwise condition, and add one or more actions to the condition.

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.

Use Case

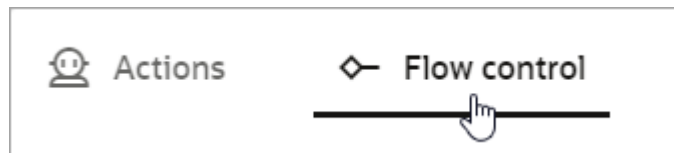
You typically add a stop within the Otherwise condition of a switch condition. For example, consider a robot that evaluates a condition. If the condition is true, the robot completes the actions in the condition and then completes the remaining actions in the robot. If the condition is false, the robot stops running. Use the stop logic to tell the robot to stop running.

Add a Stop

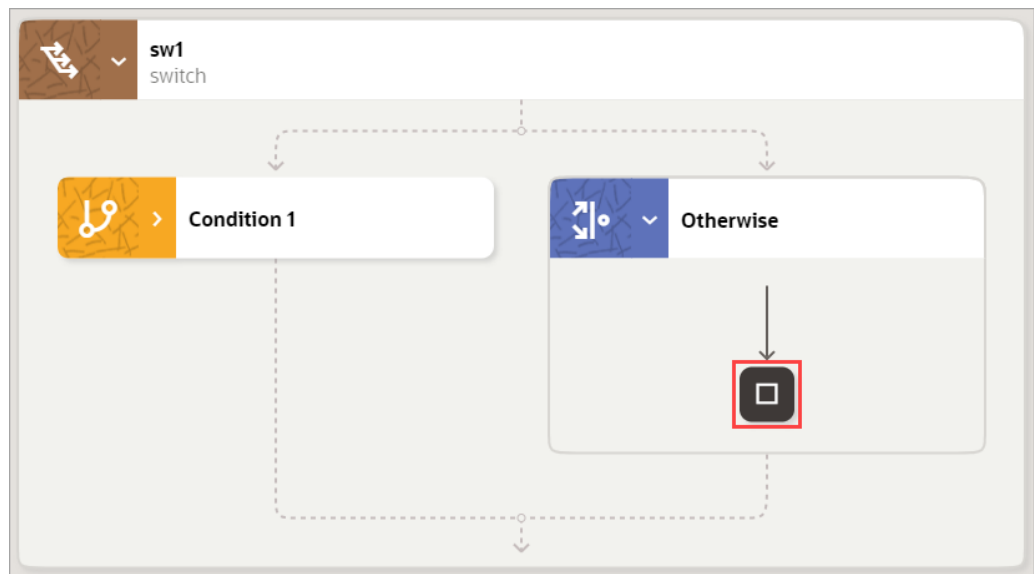
Note

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 for editing.
See [Open a Robot](#).
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**.

Fix a Robot's Errors

First, validate a robot to identify its errors. Then, determine the cause of each error and fix it.

Validate a Robot

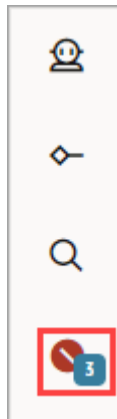
Every time you save a robot, Oracle Integration validates the robot and identifies its errors, if any exist. You can also validate a robot without saving.

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.

Validation Is Iterative

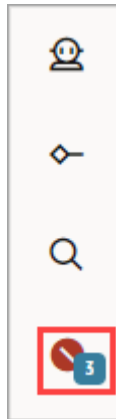
Fixing errors is often an iterative process. Try to fix errors as you build. Keep in mind that the validation process looks for errors in the following way:

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, save your robot again, and find new errors.

Fix an Error in a Robot

1. Validate a robot.
See [Validate a Robot](#).
2. On the right toolbar, select **Errors**.



The Errors panel appears.

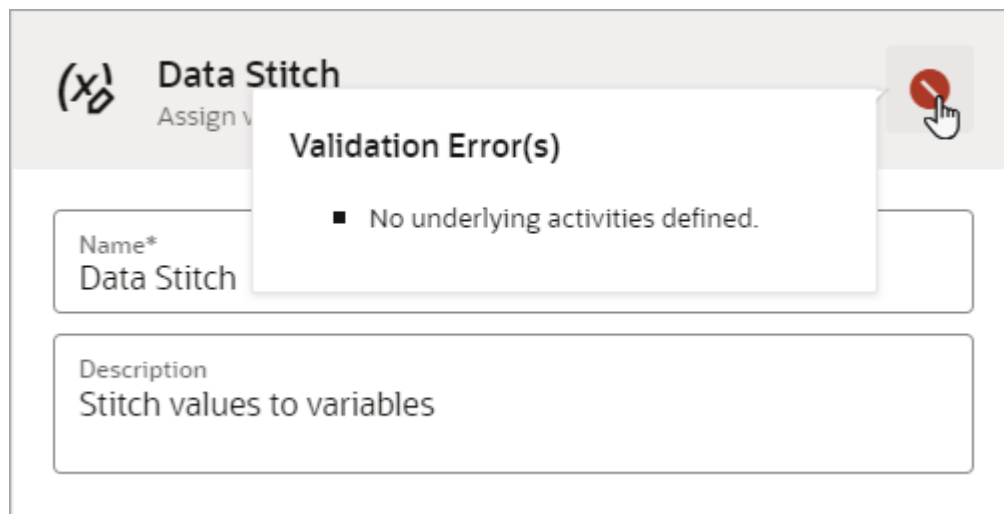
3. Point to an error, and select **Edit** .

Oracle Integration scrolls to the action on the canvas and opens its panel. 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.

Next step: [Specify Where a Robot Runs](#).

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](#)

Perform as Needed

- [Update an Environment Pool](#)
- [Remove an Environment Pool from a Robot](#)

Understand the Rules for Environments and Environment Pools

Environments are the computers or virtual machines (VMs) where robots run. Environment pools are collections of these computers. Understand the rules for setting up environments and associating robots with them.

Rule	More information
Installing the robot agent on a computer or virtual machine creates an environment	<p>A computer or VM becomes an environment when you install the robot agent on the computer or VM.</p> <p>Complete the following tasks:</p> <ol style="list-style-type: none"> 1. Meet the Robot Agent's Requirements Every environment in an environment pool should contain everything that a robot needs, such as the correct internet browser. 2. Download the Robot Agent 3. Update the Robot Agent's Configuration File 4. Install and Start the Robot Agent <p>After a computer becomes an environment, you can add the environment to an environment pool.</p>
Robots in different projects can run on the same environment pool	<p>A robot in any project can run on an environment pool that you created in any project. All you need to do is mark the environment pool as shared. See Create an Environment Pool.</p>
An environment can be part of <i>only one environment pool</i>	<p>To reduce resource conflicts, you can add an environment to only one environment pool.</p>
Change the environments appropriately for each environment	<p>A robot has the same environment pool during building and testing, as well as in production. When you promote the robot to testing or production, add and remove environments as needed.</p>

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.

Complete the following tasks:

1. [Meet the Robot Agent's Requirements](#)
2. [Download the Robot Agent](#)
3. [Update the Robot Agent's Configuration File](#)
4. [Install and Start the Robot Agent](#)


- Create the robot that must run on the environment pool.

See [Create a Robot](#).

- Familiarize yourself with environments.

See [Understand the Rules for Environments and Environment Pools](#).

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** (if you have no environment pools) or **+** (if you have one or more environment pools).

The Create environment pool panel appears.

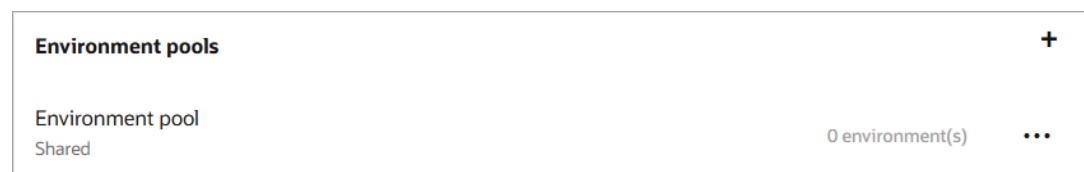
4. Fill in the fields.

Later, you'll need to associate the environment pool with a robot. Consider creating naming conventions for environment pools to simplify this task.

For **Share with other projects**, select this option if you want to allow robots in other projects to run on computers in the environment pool. If you leave this option deselected, only robots in the current project can run on computers in the environment pool.

5. Select **Create**.

If you selected **Share with other projects**, *Shared* appears below the name of the environment pool.




Next step: [Add Computers to an Environment Pool](#).

Add Computers to an Environment Pool

Group related computers or virtual machines into environment pools in a way that makes sense for your organization.

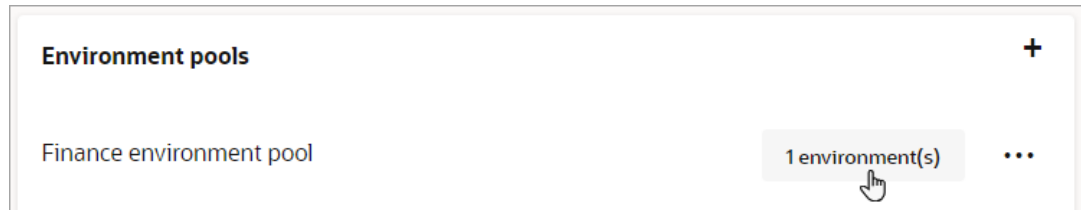
Previous step: [Create an Environment Pool](#).

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, point to an environment pool, 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. See [Install and Start the Robot Agent](#).
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.




Next step: [Associate a Robot with an Environment Pool](#).

Associate a Robot with an Environment Pool

To run a robot on a real-world environment, you must associate the robot with an environment pool.

Previous step: [Add Computers to 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**.

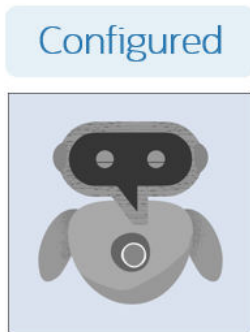
The Add environment pool panel appears.

4. Select the environment pool where the robot will run.

The list includes environment pools in the current project as well as environment pools that are shared from other projects.

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.




If the state doesn't change, you can fix the robot's errors now. See [Fix a Robot's Errors](#).

If the robot's state is Configured, you can activate the robot. See [Activate a Robot](#).

Update an Environment Pool

You can remove environments from an environment pool or rename an environment pool at any time. You typically remove environments after you've deployed a robot to a testing or production instance.

For example, an environment pool might contain your local computer while you're building the robot. Then, after you deploy a robot to a testing or production instance, you remove your local computer from the environment pool and add a different 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. To change the shared state of the environment pool, update the **Share with other project** setting as needed.

Select this option if you want to allow robots in other projects to run on computers in the environment pool. If you leave this option deselected, only robots in the current project can run on computers in the environment pool.

4. To remove an environment from an environment pool:
 - a. In the **Environment Pools** box, point to an environment pool, select **...**, and select **Remove environment**.
The Remove environment panel appears.
 - b. Select the environment(s) to remove, and select **Remove**.
5. To rename an environment pool:
 - a. In the **Environment Pools** box, point to an environment pool, select **...**, and select **Update**.
The Update environment pool panel appears.
 - b. Update the fields as needed, and select **Update**.

Remove an Environment Pool from a Robot

If a robot has a status of **Draft** or **Configured**, you can remove its environment pool. If a robot has a status of **Active**, and you need to remove its environment pool, you must deactivate the robot first.

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 **Remove environment pool**.

The environment pool is no longer associated with the robot.

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 Web Table Definitions](#)
- [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.

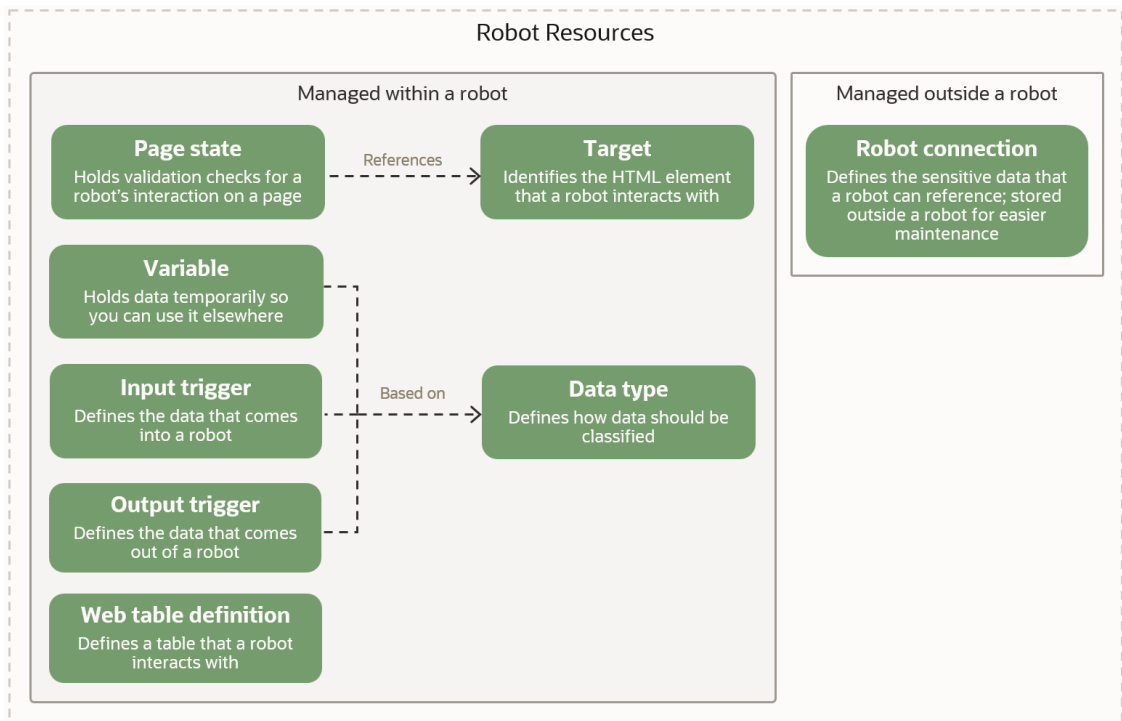
You can use many of these robot resources in a robot action. See [Define the Fields of an Action](#).

On This Page


- [Illustration](#)






- [Overview](#)
- [Web Table Definitions](#)
- [Variables](#)
- [Triggers](#)
- [Page States](#)
- [Targets](#)
- [Data Types](#)
- [Robot Connections](#)


Illustration



Overview

Option	Why they're helpful	Example
	A web table definition defines the columns in a table that a robot must interact with. Learn more: Web Table Definitions	Consider a robot that gets invoice numbers from a table that lists all invoices that meet specific criteria. You identify the table using a web table definition. When you add a web table action to a robot and select a web table definition, Oracle Integration creates a variable for saving the table information.

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. Learn more: Variables	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. Learn more: Triggers	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 output data elsewhere, such as to another robot, integration, or action in an integration. Learn more: Triggers	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.
 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. Learn more: Page States	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. Learn more: Targets	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. 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. Learn more: Data Types	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.

Option	Why they're helpful	Example
 Robot connections	A robot connection lets you store sensitive data, such as sign-in credentials, outside a robot for easier maintenance. Learn more: Robot Connections	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.

Web Table Definitions

Area	More information
How to define the values	A web table definition identifies columns in a table. You cannot hard code these columns.
Where to define the values	Create a web table definition at any time on the canvas .
How to use the values	Use a web table definition when you add a web table action to a robot.

Variables

Area	More information
How to define the values	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.
Where to define the values	Create a variable when you add an action to a robot , or at any time on the canvas .
How to use the values	If you use a variable to hold the output value of a robot action , you can use the variable however you need to. For example, 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
How to define the values	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. 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.
Where to define the values	Create an input or output in a robot, either when you define the trigger of a robot , or at any time on the canvas .
How to use the values	Use an input or output property when you define a robot action .

Page States

Area	More information
How to define the values	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.
Where to define the values	Create an input or output in a robot, either when you define the trigger of a robot , or at any time on the canvas .
How to use the values	Select a page state for a robot action when you define a robot action .

Targets

Area	More information
How to define the values	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.
Where to define the values	Create a target when you add an action to a robot , or at any time on the canvas .
How to use the values	Your settings determine whether Oracle Integration reuses targets for previously selected UI controls. You can override these settings , if needed. Reusing a target offers benefits. For example, you can update a target one time, and all actions that use the target get the update.

Data Types

Area	More information
How to define the values	A data type can contain one or more properties. Oracle Integration provides several predefined data types, including string, boolean, and number. You can define additional data types as needed, including defining their properties.
Where to define the values	Create a data type at any time on the canvas .
How to use the values	Select a variable as the basis when you create a variable or create a trigger's input or output .

Robot Connections

Area	More information
How to define the values	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. Additionally, a robot connection and its parameters live outside a robot, so you can update the parameter values without updating a robot.
Where to define the values	Create a robot connection in a project.
How to use the values	Use the value of a parameter when you define a robot action .

Work with Web Table Definitions

Create and update web table definitions on the canvas.

To learn more about web table definitions and other robot resources, see [Alternatives to Hard Coding Data](#).

Create a Web Table Definition

A web table definition defines a table that a robot interacts with.


For example, consider a robot that gets invoice numbers from a table that lists all invoices that meet specific criteria. You identify the table using a web table definition.

You can also create a web table definition while you add a web table action to a robot. See [Add a Web Table Action](#).

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 identify it using a web table definition. Instead, use the get text action and save the individual values as a variable collection. See [Add a Get Text Action](#).

2. 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.

3. In the toolbar, select **Web table definitions** .

The Web table definitions panel appears.


4. Select **Create**.
5. In the panel, enter a **Name** and **Description**.

The name appears when you add a web table action to a robot. The name should help you and others identify:

- The application where the table appears.
- The page where the table appears.
- If necessary, the table's location on the page.

6. Identify the first column of data that the robot must interact with.

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 or tab, open the application where the robot works.
- b. In the web table definition, click within the **Header** field, and select **Target a page element** .

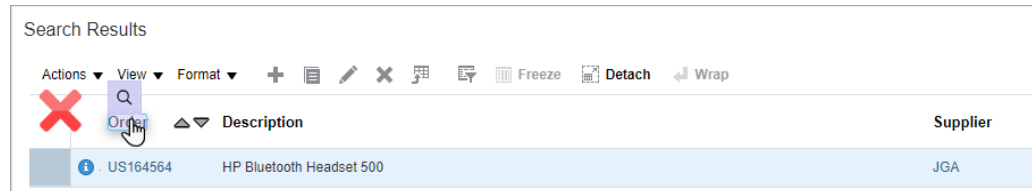
The Target a page element panel appears.

- c. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works, and select **Go**. You might need to scroll down to find the application.

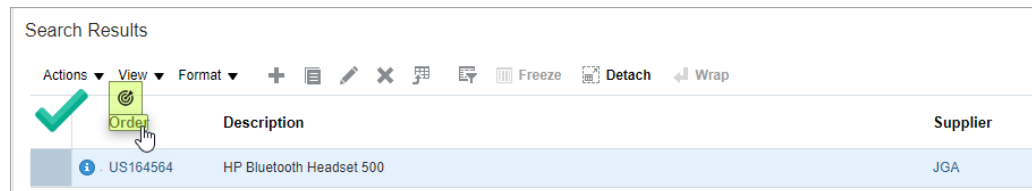
The application opens.

- d. In the application where the robot works, point to the header of the first column that the robot interacts with, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- e. Wait for the shading to turn green and the mouse icon to change to a hand, and then 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 these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions 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.
7. If the robot must interact with one or more additional columns in the table, identify the additional columns.
- Next to Columns, select **Add** **+**.
 - Repeat the previous steps to identify the header and first two rows of the column that the robot interacts with.
 - Add more columns as needed until you've identified all the columns that the robot interacts with.
8. Click **OK**.

Next, you can add a web table action to a robot. See [Add a Web Table Action](#).

Update a Web Table Definition

You can update a web table definition at any time from the canvas, without opening the actions that use the web table definition.

1. Open the robot whose web table definition 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 **Web table definitions** .
3. Select the web table definition to edit, and select **Edit** .
4. Update the web table definition as necessary, and select **OK**.
5. Above the canvas, select **Save**.

Work with Variables

Create and update variables on the canvas or in 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 in 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.

1. Determine where to start:
 - If you're currently adding an action to a robot, work within the robot action.
 - a. Add an action to a robot.
See [Add an Action to a Robot](#).
 - b. 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.
 - Otherwise, start on the canvas.
 - 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 **Variables** (x).
The Variables panel appears.
2. Select one of the following options:
 - If no variables have been created, select **Create**.
 - If at least one variable exists, 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 the checkbox is deselected, the variable can store only a single value. If a robot obtains a value and stores it in a variable, and the robot iterates over several records, you must select **Collection** so that the variable can store a value from each record.
4. Click **OK** until you return to the canvas.
5. Above the canvas, select **Save**.

Update a Variable


You can update a variable at any time from the canvas, without opening an action that uses the variable.

Before updating a variable, check where it's used: [See Where a Variable Is Used](#).

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** (x).
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**.

See Where a Variable Is Used

At any time, you can view the actions and logic that reference a variable.

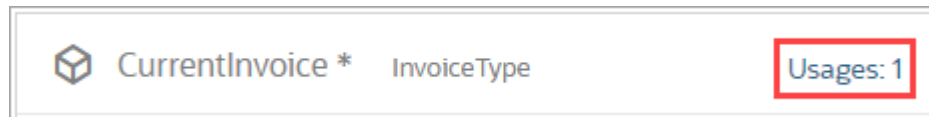
1. Open a robot.
 - a. In the navigation pane, select **Projects**.
 - b. Select the project name.
 - c. In the left toolbar, select **Robot** .
 - d. In the **Robots** box, select the robot to open.

The canvas appears.

2. On the toolbar, select **Variables** (x).

The Variables panel appears.

3. In the list, locate a variable.
4. Select the **Usages** link for the variable.



The Variable usages panel appears.

5. Review the actions and logic that reference the variable.
6. Click **OK**.

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 output 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.


2. In the toolbar, select **Trigger**  .
The Trigger panel appears.
3. Select one of the following tabs:
 - **Input:** Define a trigger's input.
 - **Output:** Define a trigger's output.
4. Define the trigger.
 - a. Select **Add**  .
 - b. Fill in the fields.
 - **Name:** Enter the name of the trigger's property. For example, an input for a person might have properties of `Name`, `Age`, and so on.
 - **Type:** Select the type on which to base the trigger. 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 the checkbox is deselected, the trigger can store only a single value.
 - c. Add additional properties as needed.
5. Click **OK**.
6. Above the canvas, select **Save**.

Update a Trigger's Input or Output

You can update a trigger at any time from the canvas, without opening 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

A page state is a container for one or more validation checks. A validation check occurs before or after an action in a robot. All the validation conditions must be met for an action to proceed.


You have a couple options for creating a page state for an action:


- While you're building a robot
 - The instructions for adding each action specify how to add page states. See [Add Validation to a Robot Action](#).
- At any time, typically before you build a robot
 - Use the instructions on this page to create a page state. Later, when you build a robot, apply the page state to a robot action and specify whether the validation occurs before or after the action.

To learn more about page states and other robot resources, see [Alternatives to Hard Coding Data](#).

Create a Page State

These instructions help you create a page state at any time.

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** (if no page states have been created) or **Create** ⁺ (if at least one page state exists).


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 Submit button enabled .

Field	Description
Message	Enter the message that appears when the validation condition isn't met before the timeout period elapses. For example, The Submit button did not become enabled within the specified 30-second timeout period. If the condition times out, the robot fails, and this message appears in the activity stream.
Strategy	Select Element : Add validation that is related to the visibility or enabled state of a UI element.
State	Select one of the following options: <ul style="list-style-type: none"> • Visible: The validation condition checks whether a UI element appears before the action proceeds. • Enabled: The validation condition checks whether a UI element is clickable before the action proceeds.
Timeout (seconds)	Enter the maximum amount of time to wait for the condition to become true. The action runs as soon as the condition becomes true. When a robot instance runs, and the time elapses before the condition becomes true, the robot instance fails.

- c. Next to Selectors, click **Create +**, and identify the UI control for the validation.

For example, if the validation ensures that a button is visible, target the button.

- i. In another browser window or tab, open the application where the robot works.
- ii. In the Page state panel, click within the **Value** field, and select **Target a page element** .

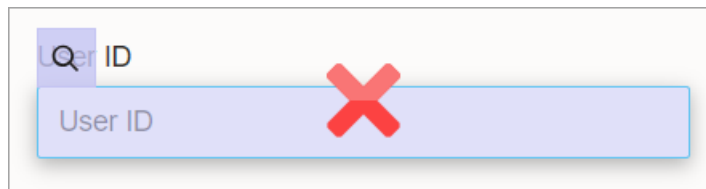
The Target a page element panel appears.

- iii. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works, and select **Go**. You might need to scroll down to find the application.

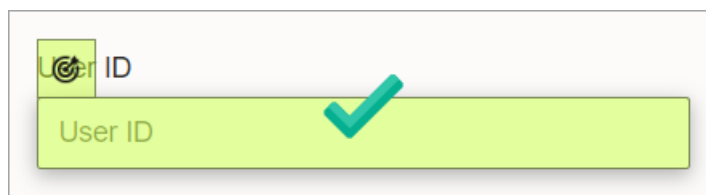
The application opens.

- iv. In the application where the robot works, point to the UI control that you're adding validation to, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- v. Wait for the shading to turn green and the mouse icon to change to a hand, and then 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 these settings](#), if needed. Reusing a target offers benefits. For example, you can [update a target](#) one time, and all actions that use the target get the update.

- d. Select **OK** in the Page state panel.
 - e. If needed, repeat the previous steps to add more page states or validation conditions.
 - f. After you finish adding page states and validation, select **OK** in the Page states panel.
6. Above the canvas, select **Save**.


Update a Page State

You can update a page state at any time from the canvas, without opening 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 **Back**.
6. Click **OK**.
7. 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.

You typically create targets while you're adding an action to a robot. Creating targets this way is straightforward. See [Add an Action to a Robot](#). You don't save any time by creating targets


before you add actions to a robot, but you can create them ahead of time if you want. Keep reading for instructions.

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** .

The Targets panel appears.

3. Select **Create** (if no targets have been created) or **Create** + (if at least one target exists).

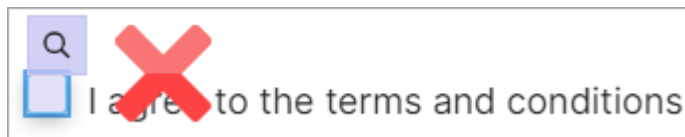
The Target panel appears.
4. Enter a name for the target.
5. Target the HTML element that the robot works with.
 - a. In another browser window or tab, open the application where the robot works.
 - b. Click within the **Locator** field, and select **Target a page element** .

The Target a page element panel appears.

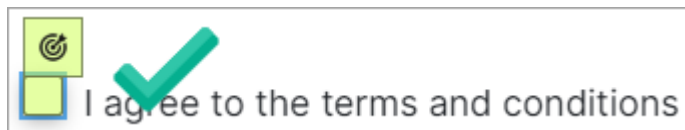
 - c. In the panel, from the **Select browser tab to target** drop-down, select the application where the robot works, and select **Go**. You might need to scroll down to find the application.

The application opens.
 - d. In the application where the robot works, point to the UI element to target, but don't select it yet.

For example, if the field is shaded purple and the magnifying lens icon appears, don't click yet. The recorder is still collecting information.



- e. Wait for the shading to turn green and the mouse icon to change to a hand, and then select the UI element.



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


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


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


Update a Target

Sometimes, the fields and buttons that a robot interacts with change. You can update a target without even opening an action that references it. A central list of all targets is right at your fingertips for stress-free maintenance.

Before updating a target, check where it's used. See [See Where a Target Is Used](#).

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. Update the target.
 - a. Above the list of targets, select **Edit**  .

The Target panel appears.
 - b. Optional: Update the fields as needed.
4. Click **OK**.
5. Above the canvas, select **Save**.

See Where a Target Is Used

At any time, you can view the actions and logic that reference 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. In the list, locate a target.
4. Select the **Usages** link on the target.



The Target usages panel appears.

5. Review the actions and logic that reference the target.

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.

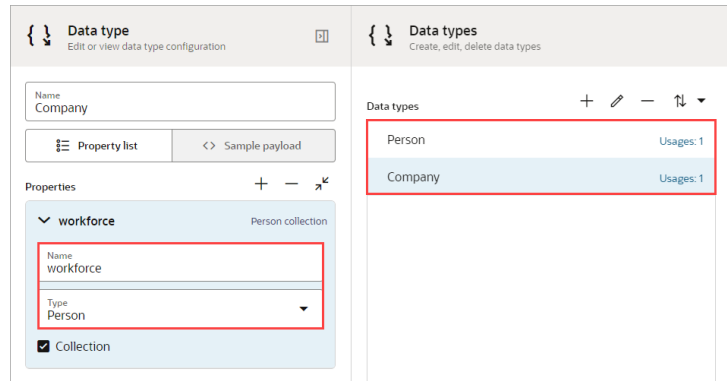
The canvas appears.
2. In the toolbar, select **Data types** .

The Data types panel appears.
3. Select **Create** (if no data types have been created) or **Create +** (if at least one data type exists).

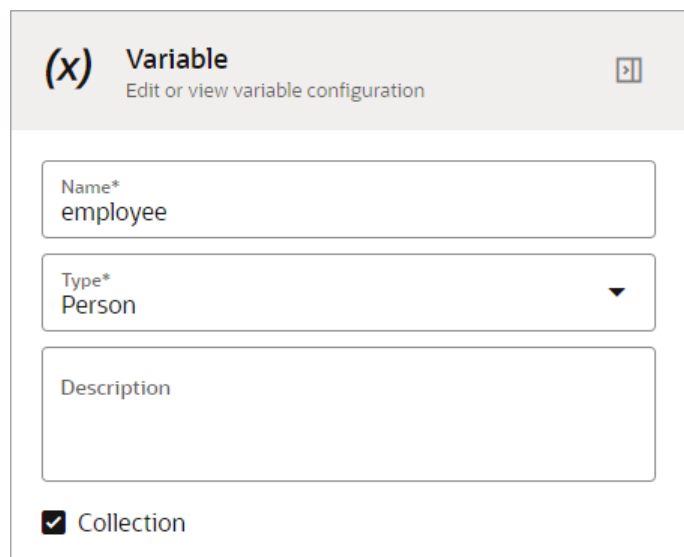
The Data type panel appears.
4. Enter the name of the data type, such as `InvoiceType`.
5. Define the data type using one of the following options:
 - Define the properties manually, one at a time.
 - a. Make sure that the **Property list** tab is selected.
 - b. Next to Properties, select **Add +**.
 - 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 the 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	Select this checkbox if the property stores an array of values. If the checkbox is deselected, the data type can store only a single value. 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. One of the properties of the <code>Company</code> data type is <code>workforce</code> , which is a collection of <code>Person</code> types.



You can create an `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. However, your update can affect a variable, input, or output that is of the data type.

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. You must correct the error before you can activate the robot. Validation happens automatically when you save, and you can also validate a robot without saving. See Validate a Robot .
Adding a property	Adding a property doesn't impact any existing robot resources or expressions.



Before updating a data type, check where it's used. See [See Where a Data Type Is Used](#).

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**.

See Where a Data Type Is Used

At any time, you can view the actions and logic that reference a data type.

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** .
 3. In the list, locate a data type.
 4. Select **Usages** link to its right.



The Data type usages panel appears.

5. Review the actions and logic that reference the data type.
6. Click **OK**.

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 instance runs, it uses the updated values.

If you need to create a robot connection, see [Create Connections to Applications](#).

Required Step After Deploying to a Higher Environment

After you deploy a robot to a higher environment, all robot connections in the project return to a Draft status, and their connection details are removed. You must update the robot connections appropriately for the higher environment. For details about deploying to a higher environment, see [Workflow for Deploying a Robot](#).


Update 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. 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've deployed a robot to a production environment, 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 robot connection, a warning appears. You can cancel or proceed. If your change makes a robot connection invalid, such as by adding a new field, the status of the robot 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	View the HTML Code for a Page
Find the XPath value for a UI element	View an Element's XPath
Identify all of the XPath values and other elements that you can target	View All Selectors for a Locator Field

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 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. The DOM also allows the recorder and other tools to interact with the page. 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.

Reasons to View an Element's XPath

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.

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

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. The DOM also allows the recorder and other tools to interact with the page. 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 XPath, paste the value 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** .
- The Target 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 left toolbar, select **Robot** .
 - d. 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 lists all the selectors, separated by commas. See [View All Selectors for a Locator Field](#).

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 example, for the following XPath: `//*[@id="buttonName"]`

Enter the following value: `xpath://*[@id="buttonName"]`

View All Selectors for a Locator Field

When you target a UI element in the Locator field of a robot action, Oracle Integration automatically captures multiple selectors for the UI element. You can view the selectors at any time.

Benefit of Capturing Multiple Selectors

Capturing multiple selectors can reduce the time that you spend troubleshooting targeting issues. For example, when a robot runs, if it can't find the first selector on a web page, the robot attempts to find the second and then third selectors.


Selectors That Are Captured

Every Locator field captures the following values, whether you target the field using the recorder or the low-code tools.




Number in the screenshot	Type of selector	Beginning of the selector value
1	Relative XPath expression	xpath://
2	Absolute XPath expression	xpath:/
3	Absolute CSS expressions	css:<tag>#

View All Selectors for a Locator Field

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. Double-click an action with a **Locator** field.

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 lists all the selectors, separated by commas, as shown in the previous screenshot.

5

Run and Test a Robot

You have several options for running and testing a robot. Each method offers benefits.

Options for Testing

Building stage and benefit	Prerequisites	Testing options
Test after you build a robot Replicate real-world conditions by testing a robot on its environment	Fix a Robot's Errors Specify Where a Robot Runs Activate a Robot	Test a Robot on Its Environment
Test after you build a robot and the integration that calls it Ensure end-to-end success by testing a robot with its integration on their environment	Design an Integration That Calls a Robot	Test a Robot and Its Integration

Workflow for Testing a Robot

Test a robot to ensure that it completes its task as expected.


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

Step	Task	More information
1	Activate a Robot	<p>After you finish building, get your robot ready to run by activating it. You must activate a robot before you can run it on its environment.</p> <p>After you activate a robot, its status changes briefly to Activation in progress, and then Active.</p> <div data-bbox="867 1318 1117 1646" data-label="Image">A green rounded rectangle with the word "Active" in white text. Below it is a square icon of a robot with a glowing yellow aura.</div>
2	Test a Robot on Its Environment	Now you can run the robot on its environments.
3	Monitor a Robot	Monitor the robot that you just ran.
4	Test a Robot and Its Integration	Replicate a real-world scenario by activating the integration and then running the robot from an integration.
5	Monitor a Robot and Its Integration	Monitor the integration and robot that you just ran.

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

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 example:

- [Add an Action to a Robot](#)
- [Add Logic to a Robot](#)
- [Fix a Robot's Errors](#)

Activate a Robot

After you finish building a robot, activate it. Afterward, you can test and run the robot.

Requirements for Activation

If version 1.0.0 of a robot is active, you can't make version 1.0.1 or version 1.1.0 active. However, you can make version 2.0.0 active. Or, you can deactivate version 1.0.0 and then activate 1.0.1 or 1.1.0.


The same rule persists as you increase version numbers. If version 2.0.0 is active, you can't make version 2.0.1 or version 2.1.0 active. However, you can make version 3.0.0 active. Or, you can deactivate version 2.0.0 and then activate 2.0.1 or 2.1.0.

Prerequisites

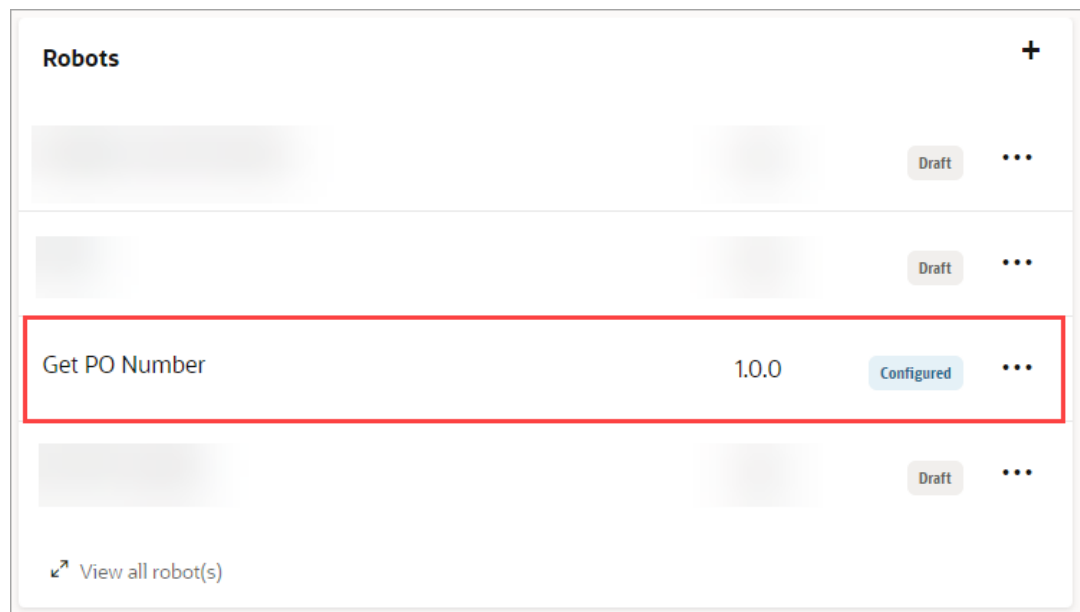
Before you activate a robot, you must complete the following steps:

- Associate the robot with an environment pool. See [Specify Where a Robot Runs](#).
- Fix all errors. See [Fix a Robot's Errors](#).

Activate a Robot

1. Open a project.
 - a. In the navigation pane, select **Projects**.
 - b. Select the project name.
2. In the left toolbar, select **Robot** .
3. Consider updating the open browser action so that the robot runs in a headless browser.

Be aware that a robot typically has improved performance when it runs in a headless browser.
4. In the **Robots** box, ensure that the robot's state is **Configured**.



If the robot isn't in a Configured state, you haven't met all the prerequisites for activation. See [Specify Where a Robot Runs](#) and [Fix a Robot's Errors](#).

- In the **Robots** box, point to the Configured robot, select ..., and select **Activate**.

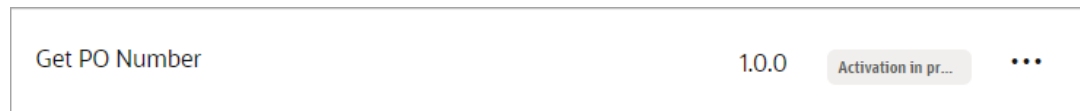
The Activate robot panel appears.

- Choose the appropriate tracing level for the robot.

See [Tracing Levels for Robots](#) and [Log Screenshots to Activity Stream](#). To understand how the different options for tracing levels impact each other, see [Options for Tracing Levels](#).

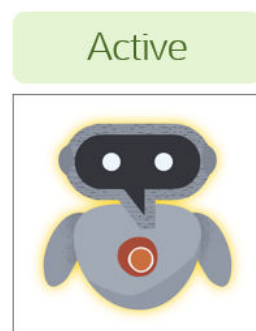
- Select **Activate**.

A Confirmation pop-up appears, and the state changes to **Activation in progress**.



- 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**.


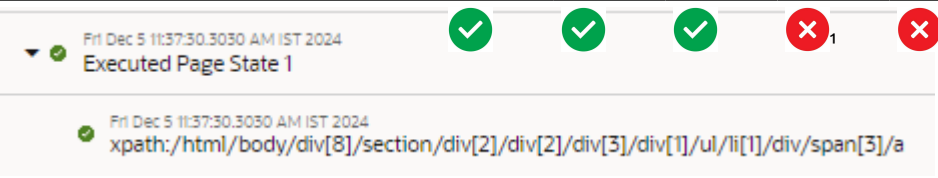



Next, you can test the robot on its environment. See [Test a Robot on Its Environment](#).

Tracing Levels for Robots

Choose the right tracing level for your robot: either production, audit, or debug.

The following table identifies the type of information that is included in the activity stream for each tracing level.

Tracing level	Example	Includes a summary of the activity	Includes all messages with unmasked contents (XPath)	Includes the input and output of activities	"Log screenshots to activity stream" is selected by default	Automatically changes to Production level after 24 hours
Production		✓	✗	✗	✗ ¹	n/a
Audit		✓	✓	✓	✗ ¹	✗
Debug		✓	✓	✓	✓	✓ ²



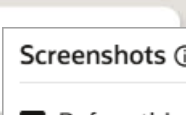



Legend

- ¹ When you select **Log screenshots to activity stream** for the Production or Audit level, screenshots are retained in the activity stream for 8 days.
- ² Data is retained in the activity stream for 24 hours. After 24 hours, minimal details about the robot instances remain available for up to 32 days. The data that remains available is the robot instance's ID, status, and time duration.

Log Screenshots to Activity Stream

When you choose a robot's tracing level, specify whether to include screenshots in the activity stream using the **Log screenshots to activity stream** checkbox.

Tip: A robot that captures screenshots sometimes runs a little slower and captures more data than a robot that doesn't capture screenshots. Therefore, many people select **Log screenshots to activity stream** for **Debug** tracing and deselect it for **Production** tracing.

Setting	Do screenshots appear in the activity stream for screenshot actions?	Do screenshots appear in the activity stream when "Before this action" or "After this action" is selected?
Log screenshots to activity stream is selected	 Screenshot Screenshot  See Add a Screenshot Action .	 Screenshots ⓘ <input checked="" type="checkbox"/> Before this action <input checked="" type="checkbox"/> After this action  See Capture Screenshots in Robots .
Log screenshots to activity stream is not selected	 See Add a Screenshot Action .	 The Log screenshots to activity stream setting overrides the Before this action and After this action settings. See Capture Screenshots in Robots .

Options for Tracing Levels

You have many options for setting tracing levels. Understand how each option affects a robot.

Option for specifying a tracing level

More information
o
e
s
t
h
i
s
o
p
t
i
o
n
a
f
f
e
c
t
s
a
r
o
b
o
t
?

Project deployment

When activating a project deployment



This setting overwrites the robot-level setting. Therefore, if you activate a robot and then activate a project deployment that contains the robot, the robot uses the tracing level that you set for the project deployment.

See [Activate or Deactivate a Project](#) in *Using Integrations in Oracle Integration 3*.

Integration activation

When activating an integration



This setting affects only an integration and doesn't affect a robot's tracing level, even if the integration calls a robot.

See [Activate an Integration](#) in *Using Integrations in Oracle Integration 3*.

Integration update

After activating an integration



This setting affects only an integration and doesn't affect a robot's tracing level, even if the integration calls a robot.

See [Manage Tracing Levels on Integrations](#) in *Using Integrations in Oracle Integration 3*.

Robot activation

When activating a robot



This setting determines the tracing level for a robot, unless you override it when you activate a project deployment.

See [Activate a Robot](#).

Robot update

After activating a robot



This setting determines the tracing level for a robot, unless you override it when you activate a project deployment.

See [Activate a Robot](#).

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

You don't need to design the integration that calls the robot.

- Specify where the robot runs and activate the robot. See [Specify Where a Robot Runs](#) and [Activate a Robot](#).
- Update your network's allowlist to allow outbound calls to Oracle Integration.
If you don't complete this task, 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](#).

Test a Robot on 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.
For help, see [Sample Code for Testing a Robot](#).
 - c. Select **Run**.
Below Instance ID, the identifier of the robot instance appears.

Next steps

Next, monitor the robot to verify that it ran as expected. See [Monitor a Robot](#).

After verifying that the robot runs successfully in its environment, test the robot with its integration. See [Test a Robot and Its Integration](#).

Sample Code for Testing a Robot

When a robot runs in the real world, an integration passes an input to a robot. However, when you run a robot independent of its integration, you must pass this value manually. To test your robot effectively, enter JSON code that matches the values that you defined in the input.

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
  }]
}
```

Test a Robot and Its Integration

After ensuring that a 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.

Workflow


1. Prerequisite: [Activate the robot](#)

If an integration tries to call a robot that isn't active, the integration fails.

2. [Activate an Integration](#)
3. [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 left toolbar, select **Integration** .
3. In the **Integrations** box, find the integration to activate.


The integration must be in a Configured state.
4. Click **Actions ...**, and select **Activate**.

The Activate integration panel opens.
5. 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 left toolbar, select **Integration** .
3. In the **Integrations** box, find the integration to run.

The integration must be in an Active state.
4. Click **Actions ...**, and select **Run**.

The Configure and run page opens.
5. Select **Run**.

The Activity Stream panel opens to show the progress of the run.

Next steps

Monitor the integration and robot. See [Monitor a Robot and Its Integration](#).

6

Deploy and Monitor a Robot

Learn how to deploy a robot to a higher environment, monitor a robot, and update a deployed robot.

What Do You Need to Do?

- [Deploy a Robot](#)
- [Monitor an Automation Solution](#)
- [Update a Deployed Robot](#)

Deploy a Robot

Learn how to deploy a robot to a test or production instance.

Topics:

- [Workflow for Deploying a Robot](#)
- [Deploy a Robot to a Higher Environment](#)

Workflow for Deploying a Robot

After you finish building a robot and testing it locally, you can deploy the robot and its integration to a higher environment, such as a test or production environment.

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

Step	Task	More information
1	Deploy your automation solution to a higher environment	You can deploy to a test, production, or other environment. The steps you follow are the same for promoting to any environment. See Deploy a Robot to a Higher Environment .
2	Update the robot connection so that it contains information for the higher environment	When you import a project deployment into a project in a new environment, the robot connections are included in the import, but their details are removed. See Update a Robot Connection .

Step	Task	More information
3	Install the robot agent on the machine(s) that the robot must run on in the higher environment	<p>Complete the following tasks for each machine that a robot must run on:</p> <ol style="list-style-type: none"> 1. Meet the Robot Agent's Requirements 2. Download the Robot Agent 3. Update the Robot Agent's Configuration File 4. Install and Start the Robot Agent 5. Start the Robot Agent Automatically (Optional)
4	Update the robot's environment pool	<p>You probably added your own computer or a development machine to the environment pool for testing and validating the robot. However, the robot might need to run somewhere else after you deploy it to a higher environment.</p> <p>The import process removed all environments from the environment pool. You must add the production machine(s) that you just configured to the environment pool. See Add Computers to an Environment Pool.</p>
5	Activate the robot	<p>After you deploy a robot to a test or production instance, the robot's status changes to Draft. Perform the following steps so that the robot can run:</p> <ol style="list-style-type: none"> 1. Remove an Environment Pool from a Robot 2. Associate a Robot with an Environment Pool <p>Associate the environment pool with the robot again so that the robot's state changes to Configured.</p> <ol style="list-style-type: none"> 3. Activate a Robot
6	Monitor your automation solution	Monitor a Robot and Its Integration

Deploy a Robot to a Higher Environment

After you finish local testing, create a deployment package that contains a robot and its integration. Import the deployment package into a higher environment, such as test or production.

Use Cases

You have several opportunities to deploy to a test or production environment:

- Many organizations deploy to a testing environment to simulate real-world usage.
- Additionally, everyone should deploy to a production environment when you're ready to go live.

Prerequisites

Before deploying to a higher environment, ensure that the robot is error free and runs successfully. See [Workflow for Building a Robot](#) and [Workflow for Testing a Robot](#).

Workflow for Deploying to a Higher Environment

1. Create a deployment package that contains the robot and its integration.
See Create and Manage a Project Deployment in *Using Integrations in Oracle Integration 3*.
2. Export the project deployment.
See Export a Project in *Using Integrations in Oracle Integration 3*.
3. Import the project deployment into the higher environment.
See Create or Import a Project in *Using Integrations in Oracle Integration 3*.

How Import Changes Your Components

The import process makes the following changes:

- The robot's status changes to **Draft**.
- The connection details in all robot connections are removed.
- All environments are removed from the environment pools.

Monitor an Automation Solution

Learn how to monitor a robot and integration together in a project, and ensure that all robot agents are up and running.

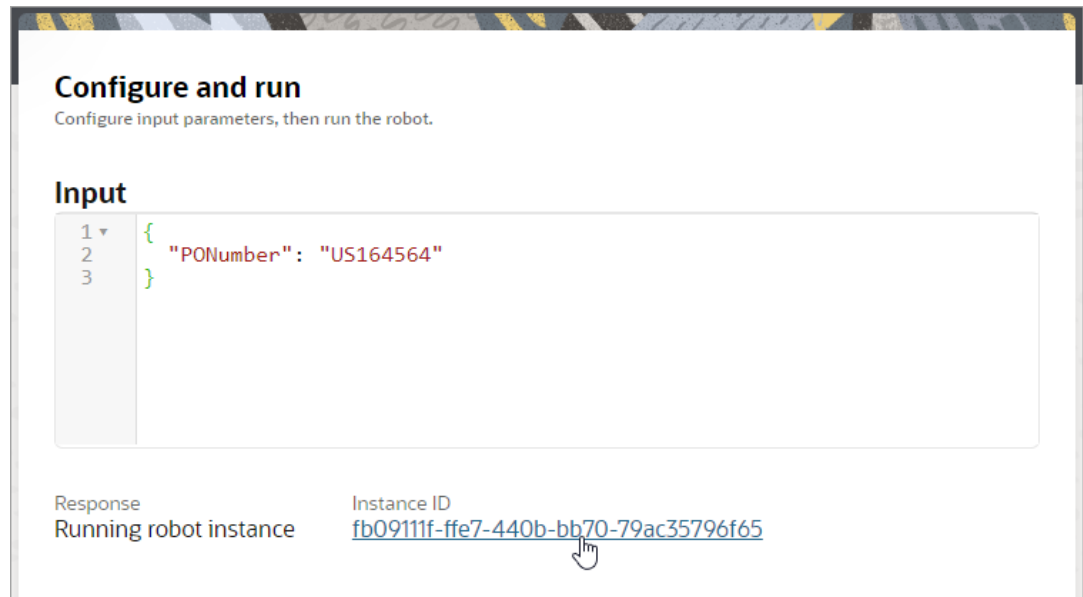
Topics:

- [Monitor a Robot](#)
- [Monitor a Robot and Its Integration](#)
- [Monitor Environments and Environment Pools](#)
- [Check Whether an Environment is Available](#)
- [Runtime Statuses of Robot Instances](#)

Monitor a Robot

Learn how to verify that a robot has run as expected.

1. Run a robot.
For details, see [Test a Robot on Its Environment](#). The steps below start where the steps for testing a robot end.
If you ran a robot by running an integration, different monitoring steps are available. See [Monitor a Robot and Its Integration](#).
2. Select the robot's Instance ID.




The Integration page opens with the Observe tab selected.

- In the left toolbar, select **Robot** .

The Robot instances tab appears.

- Check the status of the robot.

For descriptions of each status, see [Runtime Statuses of Robot Instances](#).

- Point to the robot, and select **View details** .

Robot	Instance ID	Status	Duration
Get Supplier Name 1.0.0 Finance environment pool JSmith	750614b5-284c-492b-b175-defcf1f38e19	Succeeded	Created 12 minutes ago Succeeded 11 minutes ago Duration 52 seconds

The Activity stream panel opens.

Activity stream
Instance Id: 750614b5-284c-492b-b175-defcf1f38e19

Tracing level: **Production**

- 02:55:57.5757 PM Entered <> text at xpath://*[@id="userid"]
- 02:55:57.5757 PM Entered <> text at xpath://*[@id="password"]
- 02:55:58.5858 PM Clicked element: xpath://*[@id="btnActive"]
- 02:56:02.022 PM Clicked element: xpath://*[@id="groupNode_procurement"]

- Review the activity stream for the robot.

Note

Details are available in the activity stream only after the robot has one of the following statuses: **Succeeded** or **Failed**.

- **Check whether each step ran correctly:** Green checkmarks indicate that the action completed successfully. A red X indicates that the robot wasn't able to complete the action.

> ✔ 02:55:57.5757 PM
Entered <> text at xpath://*[@id="userid"]

> ✔ 02:55:57.5757 PM
Entered <> text at xpath://*[@id="password"]

> ✔ 02:55:58.5858 PM
Clicked element: xpath://*[@id="btnActive"]

> ✔ 02:56:02.022 PM
Clicked element: xpath://*[@id="groupNode_procurement"]

- **Review the steps that the robot completed for each action:** Expand each entry in the activity stream to review the tasks that the robot performed.

✓ ✔ 01:56:28.2828 PM
Entered <> text at xpath://*[@id="password"]

▼ ✔ Thu Sep 11 01:56:28.2828 PM IST 2024
Pre-Activities

▼ ✔ Thu Sep 11 01:56:28.2828 PM IST 2024
Executed Enter Text "Password" Pre Validate1

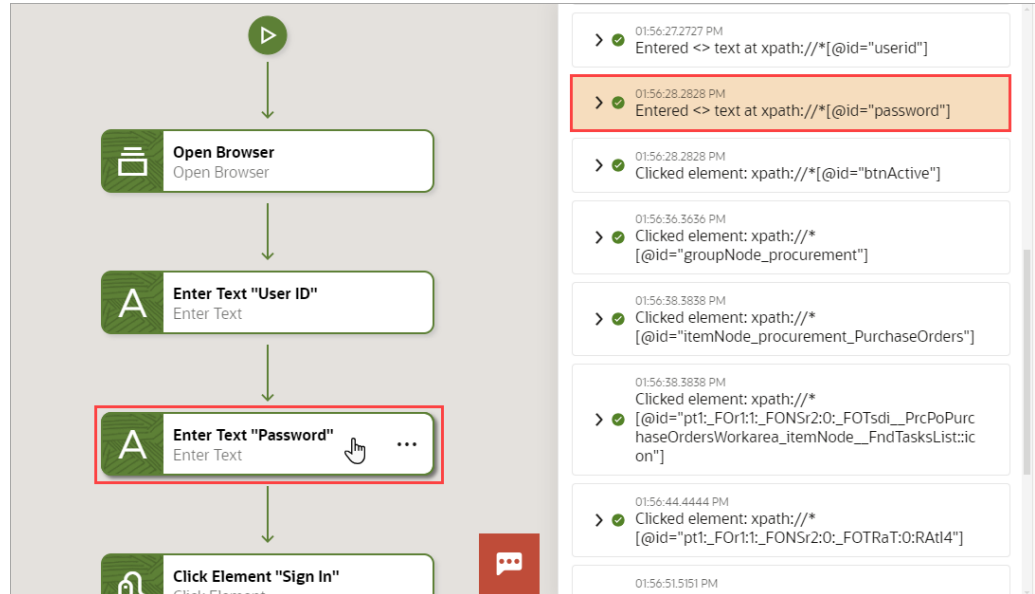
✔ Thu Sep 11 01:56:28.2828 PM IST 2024
xpath://*[@id="password"]

✔ 01:56:28.2828 PM
Entered <> text at xpath://*[@id="password"]

- **Review any information that was logged:** For example, if you added a log action to record a value in the activity stream, find and review the logged data.

✔ 02:56:15.1515 PM
Supplier name: JGA

7. Review the activity stream for the robot alongside a canvas view of the robot.
 - a. At the top of the activity stream, select the robot's instance ID.
The canvas view of the robot opens, with the activity stream on the right.
 - b. To view the activity stream entry for an action, double-click the action on the canvas.
The related record in the activity stream is highlighted in orange.



Monitor a Robot and Its Integration

Learn how to verify that an integration and robot have run as expected.


1. Run an integration that calls a robot.

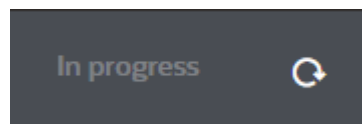
For details, see [Test a Robot and Its Integration](#). The steps below start where the steps for testing a robot and its integration end.

If you ran a robot without running its integration, different monitoring steps are available. See [Monitor a Robot](#).

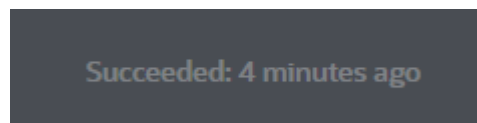
2. Monitor the integration.
 - a. Select the integration's instance ID.

The canvas view of the integration opens, with the activity stream on the right.

- b. In the upper-right corner, check the status of the integration and robot.
If the integration or robot is still running, **In progress** appears, along with a Refresh button .

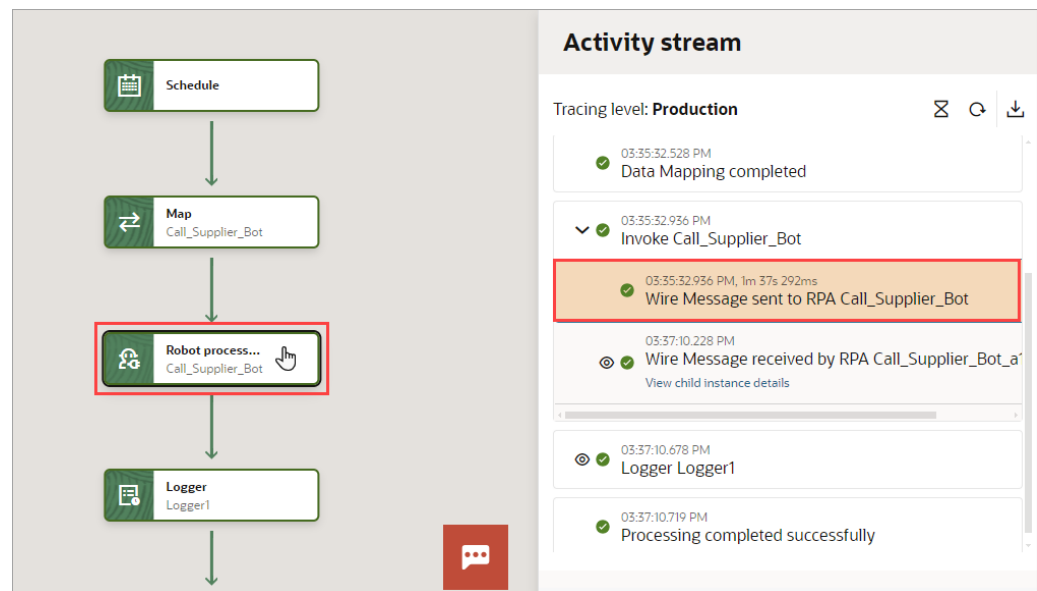


- c. Select **Refresh**  until **Succeeded** appears in the upper-right corner.

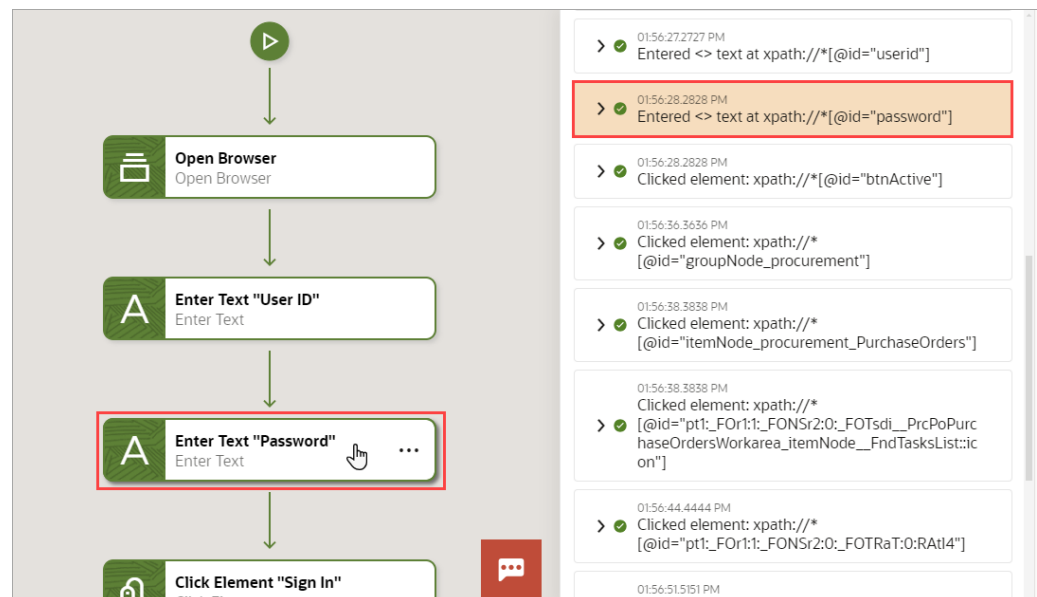


For more information about monitoring an integration, see *Monitor Integrations in a Project*.

3. Monitor the robot.
 - a. Double-click the **Robot process automation** action on the canvas.
The related record in the activity stream is highlighted in orange.



- b. Below the **Wire Message received** entry, select **View child instance details**.
The activity stream for the robot instance opens.
- c. At the top of the activity stream, select the robot's instance ID.
The canvas view of the robot opens, with the activity stream on the right.
- d. To view the activity stream entry for an action, double-click the action on the canvas.
The related record in the activity stream is highlighted in orange.



- e. Review the activity stream for the robot.
 - **Check whether each step ran correctly:** Green checkmarks indicate that the action completed successfully. A red X indicates that the robot wasn't able to complete the action.

02:55:57.5757 PM
Entered <> text at xpath://*[@id="userid"]

02:55:57.5757 PM
Entered <> text at xpath://*[@id="password"]

02:55:58.5858 PM
Clicked element: xpath://*[@id="btnActive"]

02:56:02.022 PM
Clicked element: xpath://*[@id="groupNode_procurement"]

- **Review the steps that the robot completed for each action:** Expand each entry in the activity stream to review the tasks that the robot performed.

01:56:28.2828 PM
Entered <> text at xpath://*[@id="password"]

Thu Sep 11 01:56:28.2828 PM IST 2024
Pre-Activities

Thu Sep 11 01:56:28.2828 PM IST 2024
Executed Enter Text "Password" Pre Validate1

Thu Sep 11 01:56:28.2828 PM IST 2024
xpath://*[@id="password"]

01:56:28.2828 PM
Entered <> text at xpath://*[@id="password"]


- **Review any information that was logged:** For example, if you added a log action to record a value in the activity stream, find and review the logged data.

02:56:15.1515 PM
Supplier name: JGA


Monitor Environments and Environment Pools

Monitor environments and environment pools from within a project. Here, you can identify shared environment pools, find environments that require updates to the robot agent, and check whether an environment pool has enough environments to handle its robot instance load.

1. In a project, open the **Environment pools** tab.
 - a. In the navigation pane, select **Projects**.

- b. Select the project name.
 - c. Select the **Observe** tab.
 - d. In the left toolbar, select **Robot** .
 - e. Select the **Environment pools** tab.
2. Find the information that you need on the page.

The following table provides sample questions and how to find the answers on this page.

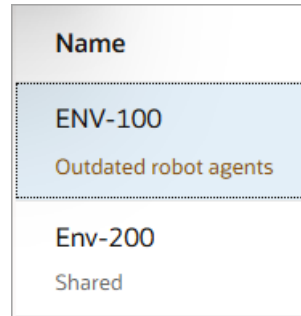
Question	How to find the answer
Which environment pools are associated with the project?	<ul style="list-style-type: none"> • Find the list of all environment pools in the table.
Which environments are in an environment pool?	<ol style="list-style-type: none"> a. In the table, select an environment pool. A page lists its environments, including their robot agent versions, host names, operating systems, and statuses. b. To view more details about each environment: Point to an environment, and in the Actions column, select Open Details .
Which environment pools are shared between projects?	<ul style="list-style-type: none"> • Find an environment pool in the table. <ul style="list-style-type: none"> – When the environment pool is shared, Shared appears below its name. – If you don't see Shared, the environment pool isn't shared.

Name
ENV-100 Outdated robot agents
Env-200 Shared

Question**How to find the answer**

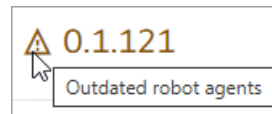
Do any robot agents need to be updated?


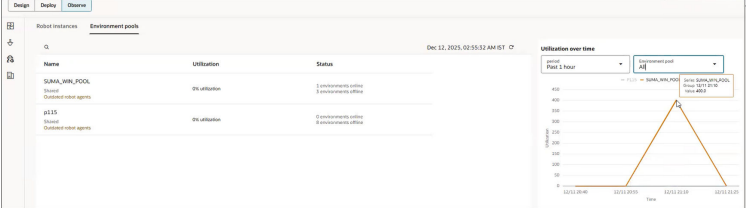
- a. Find an environment pool in the table.
- When at least one robot agent in an environment pool needs to be updated to the latest version, **Outdated robot agents** appears below the environment pool's name.
 - If this text doesn't appear, all robot agents in the pool are up to date.

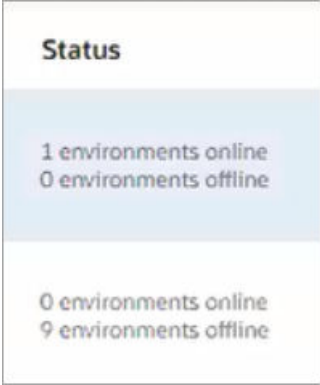


Name
ENV-100
Outdated robot agents
Env-200
Shared

- b. Select an environment pool with an outdated robot agent.
- c. In the Agent Version column, look for a robot agent with a caution symbol in front of it. When you point to the symbol, **Outdated robot agents** appears in a tooltip.



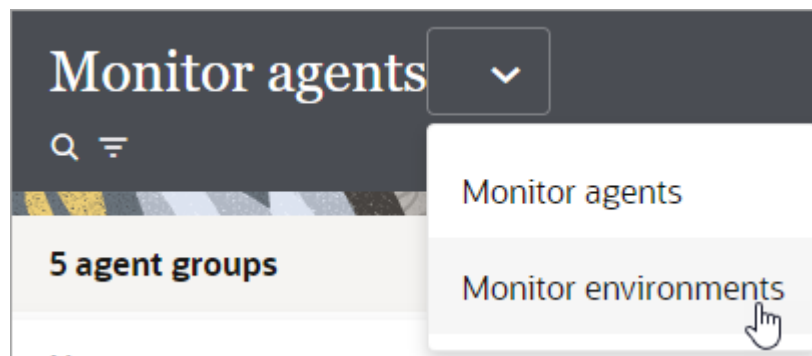
Question	How to find the answer
Do any environment pools have queued robot instances because all environments are actively running robot instances?	<ul style="list-style-type: none"> • Review the Utilization column: <ul style="list-style-type: none"> – 0% utilization indicates that no robot instances are running or queued for any environments. – 100% utilization indicates that the environment pool resources are utilized to the fullest to address the workload at the moment. – lower than 100% utilization indicates that some environment pool resources are available when processing the workload at the moment. – 101% utilization or higher indicates that at least one robot instance is in queue at that point in time because all environments are busy running other robot instances. <p>For example, if an environment pool has only one environment, and the environment is running one robot instance with two robot instances in its queue, its utilization is 300%.</p> <p>Utilization over 100% might indicate that the environment pool requires additional environments.</p>
	
	<ul style="list-style-type: none"> • Review the Utilization over time line graph details displayed in the right pane: <ul style="list-style-type: none"> – You can filter the line graph display by selecting one of the following values from the period drop-down list: Past 1 hour, Past 6 hours, Past 12 hours, Past 24 hours. – You can filter the line graph display for all environment pools or a specific environment pool by selecting the required value from the Environment pool drop-down list. – A flat line indicates 0% utilization. – Hover on the line to see the corresponding utilization values at a particular point.
	
How many robot instances are in the queue for an environment pool?	<ul style="list-style-type: none"> a. Find an environment pool for which the Utilization column shows a number above 100%, and select the row in the table. b. On the next page, view the Environment pool status section on the right, and find the Queued robot instances metric.

Question	How to find the answer
Are any environments offline in an environment pool?	<ul style="list-style-type: none"> See the Status column, which lists the number of online and offline environments in each environment pool. 
Which robots run on an environment pool?	<ol style="list-style-type: none"> Select an environment pool in the table. In the Environment pool status area on the right, select View linked robots.


Check Whether an Environment is Available

You can check the availability of all environments on a single page.

- In the navigation pane, click **Observability**, then **Agents**.
- At the top of the page, expand the **Monitor agents** drop-down, and select **Monitor environments**.



The Monitor environments page lists every environment on which the robot agent has been installed for the instance, plus the following information:

- Computer that the robot agent was installed on
 - Status of the environment
 - Environment pool that the environment is in
 - Project that the environment is associated with
- To view more information about an environment, point to the environment, and select **Expand** .
 - Check the **Status** column to identify any environments that are unavailable.

Name	Host	Status	Environment pool	Project
JSmith	JSMITH	Unavailable	Local_JSmith	FINANCE

5. Troubleshoot any environments with a status of **Unavailable**.

See [An Environment Is Unavailable](#).

Runtime Statuses of Robot Instances

A robot instance's runtime status, visible on the Observe page in a project, indicates whether the robot instance ran successfully.

Robot instance status	Description
Agent Failed	The robot instance didn't run because two or more versions of the robot agent are installed and running on an environment.
Deleted	The robot instance was deleted. Instances are deleted according to the data retention rules of the Oracle Integration instance.
Failed	The robot instance was unable to complete its task. For example: <ul style="list-style-type: none"> • An error occurred for the robot instance. • An environment wasn't available or couldn't accept the request to run the robot before the robot instance's duration exceeded the timeout limit. For details about the timeout limit, see <i>Service Limits in Provisioning and Administering Oracle Integration 3</i> . For troubleshooting help, see A Robot Fails .
In Progress	The robot instance is currently running.
New	Oracle Integration does not currently use this status for robot instances.
Pending	Oracle Integration is currently creating the robot instance or trying to assign the robot instance to an environment.
Succeeded	The robot instance completed all of its tasks successfully and has finished running.
Timeout	The robot instance timed out. For example: <ul style="list-style-type: none"> • The environment that the robot was supposed to run on was deleted. • The robot instance was still running when its duration exceeded the timeout limit. For details about the timeout limit, see <i>Service Limits in Provisioning and Administering Oracle Integration 3</i> .

Update a Deployed Robot

Learn how to update a robot that you've deployed to a higher environment, such as a production or test instance.

Topics:

- [Workflow for Updating a Deployed Robot](#)
- [Version a Robot](#)
- [Replace a Robot with an Integration](#)

Workflow for Updating a Deployed Robot

If you need to update a robot that you've deployed to a higher environment, you don't need to rerecord the robot or wade through code. The same intuitive tools that you used to build the robot are available to update the robot.

Step	Task	More information
1	Open the lower environment where you built the robot and its integration, and create a new version of the robot	Version a Robot
2	Update the robot as needed	For example: <ul style="list-style-type: none"> • Add an Action to a Robot • Add Logic to a Robot
3	Address any errors with the robot	Fix a Robot's Errors
4	Activate the robot, and test it in the build environment to ensure that the robot works as expected	Workflow for Testing a Robot
5	Deploy the robot to a higher environment	Workflow for Deploying a Robot

Version a Robot

Creating a new version of a robot gives you the ability to fall back to an earlier version of the robot. Additionally, you can have multiple versions of a robot active in an environment.

Use Cases


The following scenarios illustrate the power of versioning:

- Fall back to an earlier version of a robot

If you need to update a robot that's already been deployed to a higher environment, create a new version of the robot first. That way, if any issues occur with the new robot's version after deployment, you can always fall back to an earlier version of the robot.
- Have two or more versions of a robot active at one time

Consider the following scenario: Robot Version 1 runs successfully on an application. The application recently released an update featuring significant changes to the user interface. The changes are available in a testing environment for a couple months and then will be available in production. You create Robot Version 2, update it for the changes to the user interface, and deploy the robot to production so that you can ensure the robot runs as expected. In this situation, Robot Version 1 and Robot Version 2 are active at the same time in production.

Version a Robot

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 **Create new version**.

The Create new version panel appears.

4. Update the version field appropriately for the change.

If you must activate this robot on an instance for which another version is already active, you must increment the major version number. For example, increment from 1.0.0 to 2.0.0. Otherwise, you can increment the version number as you choose.

For guidance on how to update the robot, see [Workflow for Updating a Deployed Robot](#).

Replace a Robot with an Integration

When circumstances or requirements change, you can replace a robot with an integration without impacting your business process. Oracle Integration supports this workflow and allows you to build the way you need to, today and in the future.

Use Case

Consider a situation in which integration developers weren't available when you needed to automate a business process. You chose to move forward with the automation work by rapidly prototyping a robot and planned to replacing the robot with an integration within a year.

Replace a Robot with an Integration

- Open an integration, and replace the action that calls the robot appropriately.

For example:

- Replace the robot action with other actions.
- Design a new integration, and call it from the integration.

See [Workflow for Developing an Integration](#) in *Using Integrations in Oracle Integration 3*.

7

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 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 Robot Agent Doesn't Start
Check the availability of environments	An Environment Is Unavailable An Unused Environment Still Appears

A Robot Agent Doesn't Start

If you're unable to start 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.

How to Troubleshoot Errors

The following errors could appear in the command line or the robot agent logs.

Error	What happened	Troubleshooting steps
Agent jar version is either blank or not correct	The file name of the robot agent's JAR file no longer contains the robot agent's version number, such as 0.1.80.	Restore the file name of the JAR file to the original name, including the robot agent's version number. If you aren't sure of the correct format, you can download the agent again and copy the JAR file into your robot agent's directory. See Download the Robot Agent .
An error occurred while storing or trying to retrieve secrets from the local store. or An Exception occurred while retrieving secrets. Terminating agent	Oracle Integration was unable to write values to or retrieve values from the computer's local keystore. This error typically appears when a user doesn't have sufficient privileges. For example, one administrator might have downloaded the robot agent, and a second administrator is trying to start the robot agent. The second administrator might not be able to access the credential manager on the computer.	Run the command prompt as an administrator, and then try starting the robot agent again. See Install and Start the Robot Agent .
Could not determine the path to the JAR file. Provide full path of the agent jar.	One of the following issues occurred: <ul style="list-style-type: none"> The folder name or directory where the robot is installed contains a space. The command that you used to start the robot agent didn't include the full directory of the robot agent's JAR file. 	Check whether the folder name or directory contains a space. If so, remove the spaces from the directory, update the command to start the robot agent appropriately, and try again. If no spaces are in the folder name or directory, review the command that you used to start the robot agent. update the command so that it contains the full directory to the JAR file. See Install and Start the Robot Agent .
Either latest available version or current running agent version is null. or Either minimum supported version is not met or current running agent version is null. or Minimum supported agent version is <i>version_number</i> , current running version <i>version_number</i>	The robot agent's version is no longer compatible with Oracle Integration.	Update to a newer version of the robot agent. See Download the Robot Agent .

Error	What happened	Troubleshooting steps
<p>Error occurred while downloading dependencies like micromamba, robot runtime, rcc, etc.</p> <p>or</p> <p>Robot runtime is not available. Please check robot runtime installation before proceeding.</p>	<p>The environment on which you installed the robot agent doesn't have access to the public internet and is unable to download the required dependencies.</p>	<p>The environment must be able to access the public internet when you start the robot agent for the first time. Ensure that the computer has access to the public internet, such as by temporarily disabling the VPN, and try starting the robot agent again.</p> <p>Depending on the robot's interactions, it might need access to the public internet in general. See Design Considerations.</p>
<p>Failed creating python virtual environment for flowRT</p>	<p>This error can occur when you're trying to start the robot agent or trying to start the robot agent as a Windows service.</p> <ul style="list-style-type: none"> If you're trying to start the robot agent, one of the following issues could be occurring: <ul style="list-style-type: none"> The machine doesn't have network connectivity. The machine restricts outbound connections to the public internet. The robot agent can't create subdirectories in the user home. If you're starting the robot agent as a Windows service, you might not have defined a required environment variable. 	<p>Starting the robot agent</p> <p>When you start the robot agent for the first time, its environment must be able to access the public internet, or you must have added the required domain names or CIDR ranges to your network's allowlist. Perform one of the following steps:</p> <ul style="list-style-type: none"> Ensure that the computer has access to the public internet, such as by temporarily disabling the VPN. Update your network's allowlist so that the environment can access the required websites. See Update Your Network's Allowlist. <p>Additionally, run the command prompt as an administrator. Then, try starting the robot agent again. See Install and Start the Robot Agent.</p> <p>When you start the robot agent again in the future, the agent might continue needing access to the public internet. See Design Considerations.</p> <p>Starting the robot agent as a Windows service</p> <p>See Start the Robot Agent Automatically.</p>
<p>Failed saving the modified InstallerProfile configuration file</p>	<p>Oracle Integration was unable to update the robot agent's configuration file. Oracle Integration must remove sensitive values from the configuration file and add a comment to the file about the change.</p>	<p>Run the command prompt as an administrator, and then try starting the robot agent again. See Install and Start the Robot Agent.</p>

Error	What happened	Troubleshooting steps
Failed to create environment in the control room or update <code>Agent.properties</code> file.	An issue occurred during the installation of the robot agent. For example: <ul style="list-style-type: none"> The installation parameters for the robot agent weren't correct. The robot agent couldn't register with Oracle Integration. The <code>Agent.properties</code> file couldn't be updated. 	Confirm that your network allows the computer to access Oracle Integration. See Review Your Network Configuration . Next, confirm that your network allows the computer to access the public internet. This access is required the first time that you start the robot agent. See Install and Start the Robot Agent . Finally, ensure that the person who starts the robot agent has the require file system permissions to save files on the computer.
Format of minimum supported version or current running agent version is not correct.	The file name of the robot agent's JAR file no longer contains the robot agent's version number, such as <code>0.1.80</code> .	Restore the file name of the JAR file to the original name, including the robot agent's version number. If you aren't sure of the correct format, you can download the agent again and copy the JAR file into your robot agent's directory. See Download the Robot Agent .
Got an exception while generating IDCS Access Token	An issue has occurred with the credential manager. For example: <ul style="list-style-type: none"> In the <code>InstallerProfile.cfg</code> file, you might not have specified a value for <code>IDCS_CLIENT_ID=</code> or <code>IDCS_CLIENT_SECRET=</code>, or the value was incorrect. The person who is starting the robot agent doesn't have access to the credential store. 	Verify the values Verify the values of <code>IDCS_CLIENT_ID=</code> or <code>IDCS_CLIENT_SECRET=</code> that you used in the <code>InstallerProfile.cfg</code> file. If the values were missing or invalid, perform the following steps: <ol style="list-style-type: none"> Download the robot agent, and copy the <code>Agent.properties</code> and <code>InstallerProfile.cfg</code> files to the directory where you've installed the robot agent. See Download the Robot Agent. Update the Robot Agent's Configuration File Install and Start the Robot Agent Start the robot agent as an administrator Run the command prompt as an administrator (or with <code>sudo</code> privilege), and try starting the robot agent again. See Install and Start the Robot Agent .
<code>InstallerProfile.cfg</code> or <code>Agent.properties</code> file is either missing fields or has invalid data	The <code>InstallerProfile.cfg</code> file or <code>Agent.properties</code> file has been modified and is no longer valid.	<ol style="list-style-type: none"> Download the robot agent, and copy the <code>Agent.properties</code> and <code>InstallerProfile.cfg</code> files to the directory where you've installed the robot agent. See Download the Robot Agent. Update the Robot Agent's Configuration File Install and Start the Robot Agent

Error	What happened	Troubleshooting steps
No value passed to -jar argument for agent jar. Wrong arguments passed or Unable to retrieve the -jar argument value. Wrong arguments passed	The command that you used to start the robot agent was missing one or more parameters. For example, the command might not contain a required parameter, or the command contains an environment variable that hasn't been set in the command prompt.	Update the command so that it contains the correct parameters. See Install and Start the Robot Agent .
RPA Agent is supported on JDK version 1.17 and above, please install or upgrade JDK/JVM to run RPA Agent.	Java Development Kit (JDK) 17 isn't installed on the computer or virtual machine.	Install Java Development Kit (JDK) 17 on the computer or virtual machine. See Meet the Robot Agent's Requirements .

Error	What happened	Troubleshooting steps
<p>User agent set to: Oracle-JavaSDK/2.56.0 (Mac OS X/14.3.1; Java/17.0.8; JavaHotSpot(TM) 64-Bit Server VM/17.0.8+9-LTS-211)</p> <p>Exception in thread "main" com.oracle.bmc.model.BmcException: Error returned by GetAgentVersion operation in Agent service. (401, NotAuthenticated, false) The required information to complete authentication was not provided or was incorrect. (opc-request-id: <i>debugging_identifier/debugging_identifier/debugging_identifier</i>)</p> <p>Timestamp: <i>timestamp</i></p> <p>Client version: <i>robot_agent_version</i></p> <p>Request Endpoint: <i>endpoint</i></p> <p>Troubleshooting Tips: See https://docs.oracle.com/en-us/iaas/Content/API/References/apierrors.htm#api</p>	<p>An issue has occurred with the <code>Agent.properties</code> file.</p>	<p>Navigate to the folder that contains the unzipped files for the robot agent, and delete the <code>Agent.properties</code> file.</p> <p>Next, try starting the agent again.</p>

Error	What happened	Troubleshooting steps
errors_401_401_n	<p>otauthenticated for more information about resolving this error Also see for details on this operation's requirements. To get more info on the failing request, you can enable debug level logs as mentioned in `Using SLF4J for Logging section` in https:// docs.oracle.com/e n-us/iaas/ Content/API/ SDKDocs/ javasdkconfig.htm .</p> <p>If you are unable to resolve this Agent issue, please contact Oracle support and provide them this full error message. at com.oracle.bmc.ht tp.internal.Respo nseHelper.throwIf NotSuccessful(Res ponseHelper.java: 160)</p>	

What to Do if an Error Doesn't Occur

After completing each step, try running the agent again.

Step	Consideration	More information
1	Does the robot agent's computer meet the system requirements for the robot agent?	For example, the correct version of the JDK might not be installed, or the JAVA_HOME might not be set. See System Requirements .
2	Does the robot agent's computer have internet access?	Navigate to a website that your network allows you to access, and confirm that the page loads. If the computer doesn't have internet access, work with a network administrator to correct this issue.
3	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. If you are unable to sign in, your network might restrict access to the internet based on IP address. 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. See Obtain the Inbound and Outbound IP Addresses of the Oracle Integration Instance in <i>Provisioning and Administering Oracle Integration 3</i> .
4	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. See Ensure that the Confidential Application is Active .
5	Is the robot agent's configuration file defined correctly?	For step-by-step instructions on updating the file, see Update the Robot Agent's Configuration File .
6	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. Additionally, Oracle recommends not including any spaces in the directory path.
7	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 Download the Log File for a Robot or Robot Agent .

An Environment Is Unavailable

When an environment's status is **Unavailable** on the Monitor environments page, open the command line from which you started the robot agent for the environment. Next, find the error that occurred, and follow the troubleshooting workflow to get the environment back up and running.

Check Whether an Environment Is Available

See [Check Whether an Environment is Available](#).

Errors and How to Address Them

You might encounter one or more of the following errors in the command line or the robot agent logs when an environment is unavailable. Use the troubleshooting steps to get back on track.

Error	What happened	Troubleshooting steps
An error occurred while storing or trying to retrieve secrets from the local store. or An Exception occurred while retrieving secrets. Terminating agent	Oracle Integration was unable to write values to or retrieve values from the computer's local keystore. This error typically appears when a user doesn't have sufficient privileges. For example, one administrator might have downloaded the robot agent, and a second administrator is trying to start the robot agent. The second administrator might not be able to access the credential manager on the computer.	Run the command prompt as an administrator, and then try starting the robot agent again. See Install and Start the Robot Agent .
Either minimum supported version is not met or current running agent version is null.	The robot agent's version is no longer compatible with Oracle Integration.	Update to a newer version of the robot agent. See Download the Robot Agent .

An Unused Environment Still Appears

After you uninstall the robot agent on an environment, the environment sometimes still appears in Oracle Integration. Delete the environment from the Observability pages.

Issue

You uninstalled the robot agent from an environment, but the environment still appears in Oracle Integration.

What to Do

1. In the navigation pane, click **Observability**, then **Agents**.
2. From the **Monitor agents** drop-down at the top of the page, select **Monitor environments**.
3. In the list, find the environment to delete.

✓ Tip

Each name is an identifier that is generated in the `Agent.properties` file.

4. Point to the row, select **Actions** , and then select **Delete**.

Troubleshoot the Recorder

Get help when the screen recorder that you use to build a robot doesn't work the way that you expect.

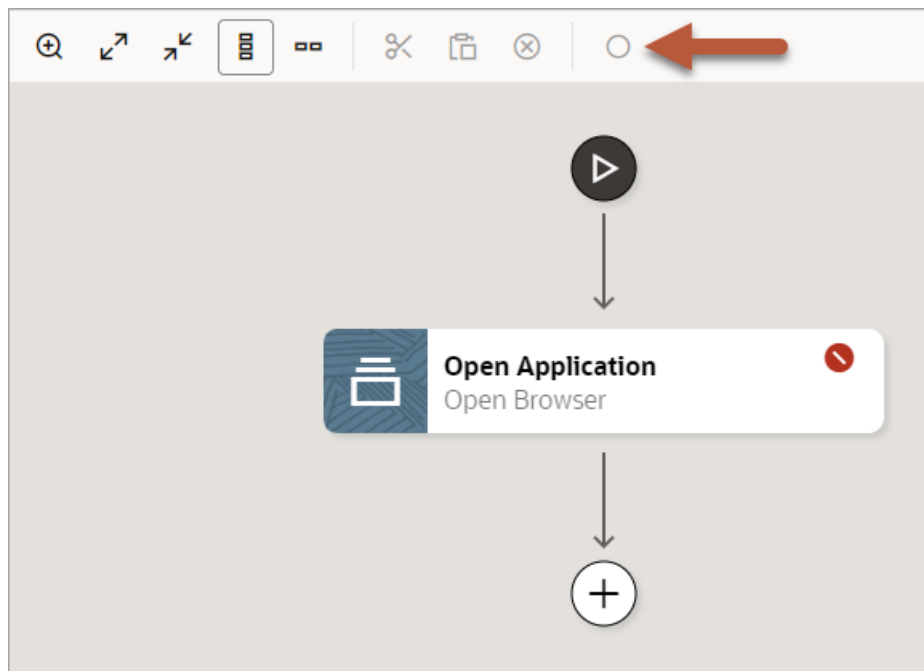
Goal	Troubleshooting help
Start the recorder	The Recorder Doesn't Start
Build a robot using the recorder	The Recorder Stops Working An Application Isn't in the Browsers List The Canvas Is Read-Only The Recorder Doesn't Capture the Correct Element for an Image An Error Occurs When Targeting an Inline Frame

The Recorder Doesn't Start

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.

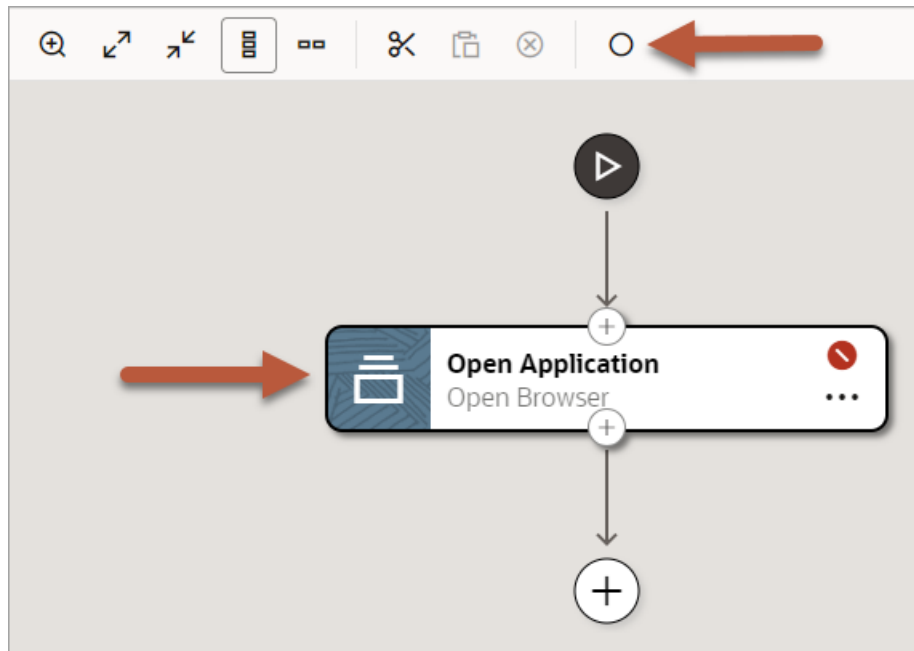


Why It Happens

The button becomes enabled only when you select an action in the canvas.

What to Do

Select an action in the canvas.



The Recorder 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.

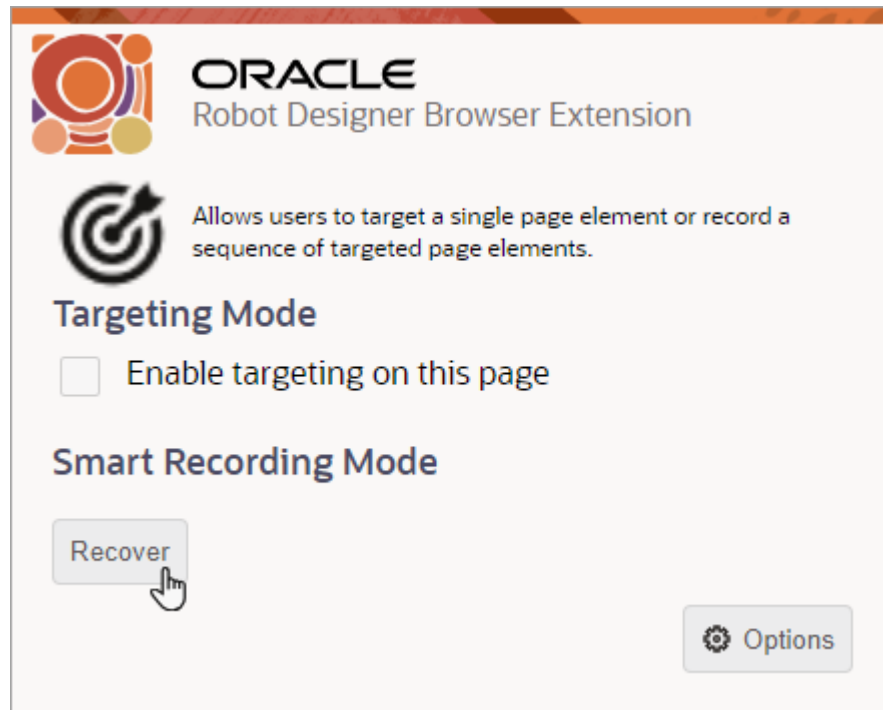
What to Do

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.

An Application Isn't in the 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.

The Canvas Is Read-Only

If everything is read-only, you're probably still recording.

Issue

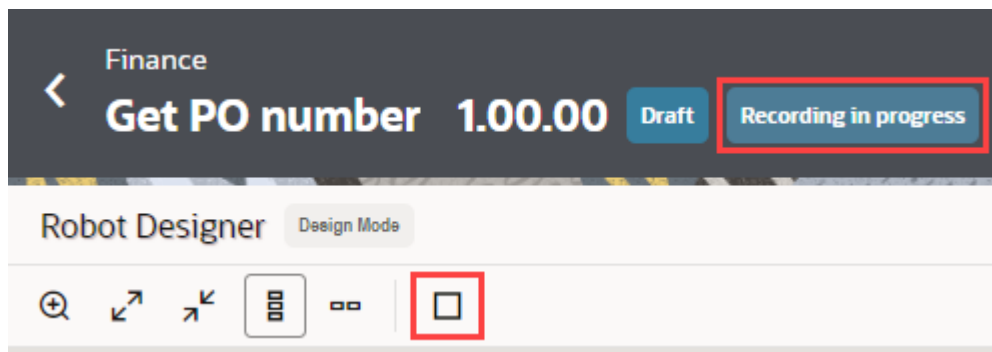
Most actions on the canvas are read-only or unavailable.

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.

The Recorder Doesn't Capture the Correct Element for an Image

When a robot fails because the recorder doesn't capture the correct element for an icon or other image, you can use the built-in tools in Google Chrome to find the correct element.




Issue

A robot fails because the recorder wasn't able to capture the correct element for a graphical element, such as an icon.

Why It Happens

For some images with `svg` or `img` tags, the recorder can't identify the ID. You must target the ID, which is the element that an action acts on.

For example, consider the following options for targeting an icon and the elements that they target. None of these options records the element's ID.

Targeting option	Element that is targeted
	<div data-bbox="678 310 898 348">Action Details</div> <div data-bbox="678 373 753 401">Name</div> <div data-bbox="683 411 1425 474" style="border: 1px solid #ccc; padding: 2px;">Click Element</div> <div data-bbox="678 520 829 552">Target name ◇</div> <div data-bbox="683 562 1425 625" style="border: 1px solid #ccc; padding: 2px;">Target</div> <div data-bbox="678 638 781 669">Element</div> <div data-bbox="683 680 1425 743" style="border: 1px solid #ccc; padding: 2px; border-color: green;">TD</div>
	<div data-bbox="678 842 898 879">Action Details</div> <div data-bbox="678 905 753 932">Name</div> <div data-bbox="683 942 1425 1005" style="border: 1px solid #ccc; padding: 2px;">Click Element "Notifications (2 unread)"</div> <div data-bbox="678 1052 829 1083">Target name ◇</div> <div data-bbox="683 1094 1425 1157" style="border: 1px solid #ccc; padding: 2px;">Notifications_2_unread</div> <div data-bbox="678 1169 781 1201">Element</div> <div data-bbox="683 1211 1425 1274" style="border: 1px solid #ccc; padding: 2px; border-color: green;">path</div>
	<div data-bbox="678 1373 898 1411">Action Details</div> <div data-bbox="678 1436 753 1463">Name</div> <div data-bbox="683 1474 1425 1537" style="border: 1px solid #ccc; padding: 2px;">Click Element "Notifications (2 unread)"</div> <div data-bbox="678 1583 829 1614">Target name ◇</div> <div data-bbox="683 1625 1425 1688" style="border: 1px solid #ccc; padding: 2px;">Notifications_2_unread</div> <div data-bbox="678 1701 781 1732">Element</div> <div data-bbox="683 1743 1425 1806" style="border: 1px solid #ccc; padding: 2px; border-color: green;">svg</div>

What to Do

1. Choose any of the incorrect targeting options in the recorder.

2. Pause the recorder.
3. In the application for which you're building a robot, right-click the element you're trying to target, and select **Inspect**.

The DevTools appear.

4. On the **Elements** tab, find the element that wraps the `svg` or `img` tag. This element contains a `id=`. For example:

```

<tbody>
  <tr>
    <td>
      <a id="pt1:_UISatr:0:cill1" title="Notifications (2 unread)" class="svg-glob xko_p_AFIconOnly" onclick="this.focus();return false;" href="#"> == $0
        <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 28 28" class="svg-bkgd06 xi6" width="24" height="24" id="pt1:_UISatr:0:cill1:icon" role="img" aria-label="Notifications (2 unread)" _afsvg="true">
        </a>
      </td>
    </tr>
  </tbody>

```

5. Copy the value that appears within the quotation marks for `id=" "`.
6. Test that you've identified the correct element.
 - a. In DevTools, select the **Console** tab.
 - b. Type the following value, and press **Enter**:


```
document.getElementById('<value_you_copied>')
```
 - c. Type the following value, and press **Enter**:


```
document.getElementById('<value_you_copied>').click()
```

If the menu opens for the icon, you've targeted the correct element.
 - d. If the `id=" "` value is no longer in your clipboard, copy it again.
7. In Oracle Integration, open the target for editing.

See [Update a Target](#).
8. Update the **Locator** value. For instance, paste the value that you copied between the quotation marks in the `@id=" "` value.

In the **Locator** field, if any additional text appears after the closing bracket `]`, remove the extra text. For example, the value should be something like this:

```
xpath://*[@id="pt1:_UISatr:0:cill1"]
```
9. Try running the robot again.

An Error Occurs When Targeting an Inline Frame

When an error occurs while targeting a frame or its contents, specify the XPath value manually.

Issue

An error occurs when you use the low-code tools to target an `iframe` or an element in an `iframe`. The error looks like this:



Why It Happens

Depending on how a website is set up, the frame action might or might not be able to target the content in a frame. For example, if an inline frame has restrictions on accessing cross-origin resources, the frame action cannot capture an XPath value for the inline frame and its elements.

What to Do

1. Update the **Locator** field for the frame action.
 - a. Open the frame action for editing.
 - b. If a value exists in the **Locator** field, clear the value.
 - c. Type the following value in the **Locator** field:

xpath:

- d. Copy the XPath value of the parent iframe, and paste it into the **Locator** field after the xpath: value.

If you're not sure how to copy the XPath value from a web browser, see [View an Element's XPath](#).

For example, the Locator value might look something like this:

xpath://*[@id="Multiple"]/iframe

- e. Type the following value immediately after the XPath value in the **Locator** field:

,xpath:

For example:

xpath://*[@id="Multiple"]/iframe,xpath:

- f. Copy the XPath value of the embedded iframe, and paste it into the **Locator** field immediately after the ,xpath: value.

For example, the entire Locator value might look something like this:

xpath://*[@id="Multiple"]/iframe,xpath:/html/body/section/div/div/iframe

2. Update the **Locator** fields for all actions that are in the frame action.

If an action doesn't contain a Locator field, you don't have to perform these steps for the action.

- a. Open an action that is in the frame action.
- b. If a value exists in the **Locator** field, clear the value.
- c. Type the following value in the **Locator** field:

xpath:

- d. Copy the XPath value of the UI element that the action interacts with, and paste it into the **Locator** field immediately after the `xpath:` value.

If you're not sure how to copy the XPath value from a web browser, see [View an Element's XPath](#).

For example, the Locator value might look something like this:

```
xpath:/html/body/section/div/h5
```

- e. Repeat the steps for any other actions with a **Locator** value.

Troubleshoot Robots

Get help troubleshooting robots.

What Are You Trying to Do?

Goal	Troubleshooting help
Activate a robot	A Robot Doesn't Activate
Add a robot	Robot Page Is Blank in a Project
Run a robot	A Robot Fails

A Robot Doesn't Activate

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](#).

A Robot Fails

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	<p>Credentials without access</p> <p>The robot doesn't have the required access</p>	<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 example, 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>	<p>Verify that the robot's credentials have the required access to complete its tasks.</p>
2	<p>Environment pool issues</p> <p>The environment pool that the robot is associated with doesn't have any environments, or none of its environments are active</p>	<p>A robot must run on an environment.</p>	<p>If the environment pool has no environments, add one or more environments to it. See Specify Where a Robot Runs.</p>
3	<p>Timeout</p> <p>The robot timed out</p>	<p>A robot can time out in the following ways:</p> <ul style="list-style-type: none"> The total running time for the robot exceeds the service limit. This timeout typically occurs due to a programming error. For example, the robot might create an infinite loop. To address this issue, determine the action where the robot got stuck, and make the necessary corrections. 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 Add Validation to a Robot Action. 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 Add a Wait Until Element Is Visible Action. 	<p>You might or might not need to address timeouts. For example, 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 example, increasing the timeout period for a single action might prevent robot instances from timing out.</p>

Step	Why it happens	More information	What to do
4	Changed HTML 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 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 View an Element's XPath .
5	Incorrect action used 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 example, sometimes a UI field looks like a drop-down list but is actually coded as a text field. For example, the following actions are used for similar activities: <ul style="list-style-type: none"> • Add a Checkbox Action • Add a Click Element Action • Add an Enter Text Action • Add a List Action 	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	Incorrect UI element targeted 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.	Targeting an image If you are targeting an icon or other image, see The Recorder Doesn't Capture the Correct Element for an Image . Targeting other elements Use the advanced settings for the recorder to view all the elements to target before saving a specific XPath. See View All Selectors for a Locator Field . 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 View an Element's XPath .
7	Dynamically generated UI elements The targets that you captured are no longer valid because the application generates targets for its UI elements dynamically	Some applications generate targets dynamically. For example, a field's target is ABC in one session and DEF in another session. The robot fails with the following error: The target element is not visible.	Use the advanced settings for the recorder to target a value that isn't dynamically generated. See View All Selectors for a Locator Field .

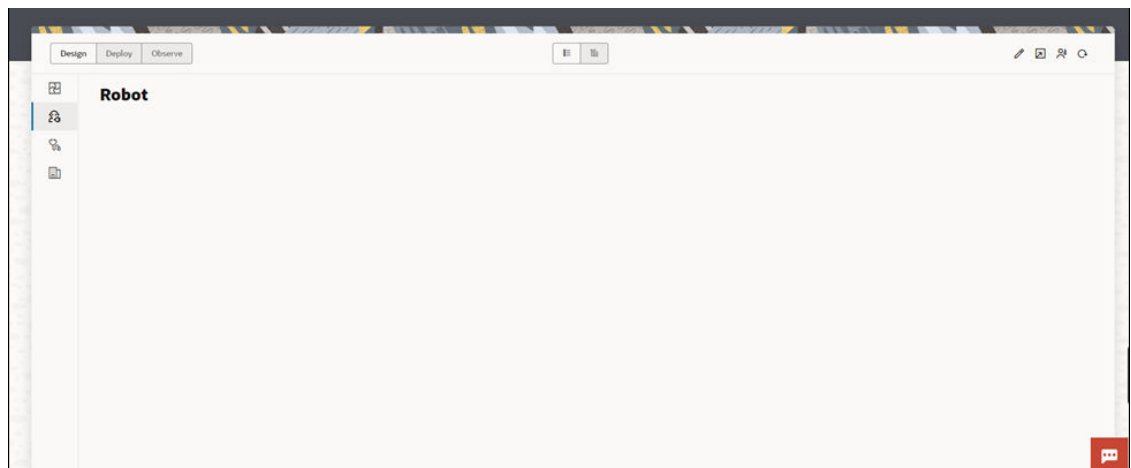
Step	Why it happens	More information	What to do
8	<p>Browser issues</p> <p>The browser that you selected isn't installed on the environment that the robot is running on, or the robot is unable to run in a non-headless browser</p>	<p>One of the following errors appears in the activity stream when the robot tries to run:</p> <ul style="list-style-type: none"> BrowserNotFoundError : Failed to start a browser: - chrome: Could not get browser version BrowserNotFoundError : Failed to start a browser: - firefox: Could not get browser version 	<ol style="list-style-type: none"> 1. Ensure that the browser you selected in the open browser action is installed on the environment. See Add an Open Browser Action. 2. Run the robot in a headless browser instead. Specify this option in the open browser action. See Add an Open Browser Action.
9	<p>Page synchronization issues</p> <p>Due to page synchronization issues, buttons aren't visible when expected</p>	<p>When page synchronization issues cause a robot to fail, an error appears in the activity stream. For example:</p> <pre>ElementClickInterceptedException: Message: element click intercepted: Element <target_element> is not clickable at point (x, y). Other element would receive the click: <other_element></pre>	<p>Fix the issue by adding a wait action before the action that fails. Start with a small wait times, such as 10 seconds, and work upward to higher wait times if the robot continues to fail.</p> <p>See Add a Wait Action.</p>

Robot Page Is Blank in a Project

If the Robot page in a project is blank, an administrator must update some security settings for the identity domain in OCI Identity and Access Management (OCI IAM).

Issue

When you open a project, the Robot page is blank. Boxes for robots, environment pools, robot connection types, and robot connections don't appear.



Why It Happens

The sign-on policy for the identity domain contains a sign-in rule with the following settings:

- **Prompt for reauthentication** is selected.
- **Every time** is selected for **Frequency**.

What To Do

Complete these steps in OCI IAM.

1. Open the default sign-on policy for the identity domain, and open the sign-on rule so that you can edit it.

See [Updating a Sign-On Policy](#) in the Oracle Cloud Infrastructure Documentation.

2. Make the following updates:

- Deselect **Prompt for reauthentication**.
- For the **Frequency** field, select **Once per session or trusted advice**.

If you don't want to make these change in the default sign-on policy for the identity domain, create a custom sign-on policy that specifies these settings. Link this policy to the Oracle Integration service instance. See [Creating a Sign-On Policy](#) in the Oracle Cloud Infrastructure Documentation.

3. If the Robots page is still blank after making these updates, file a service request.

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 on Invoices](#)
- [Use Case: Save Values After Iterating on a Table](#)
- [Use Case: Switch Browsers](#)
- [Use Case: Switch Windows](#)

Use Case: Update a Set of Invoices

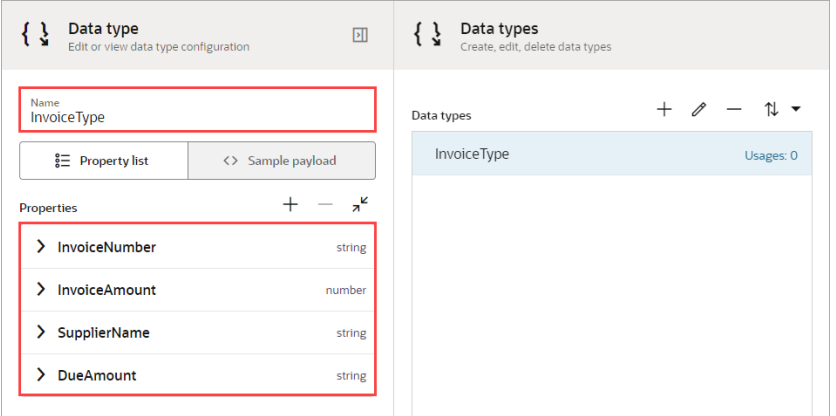
In this use case, an organization must manually update a set of invoices. Explore how data types, triggers, a foreach loop, a switch condition, and expression syntax support this workflow.

Scenario

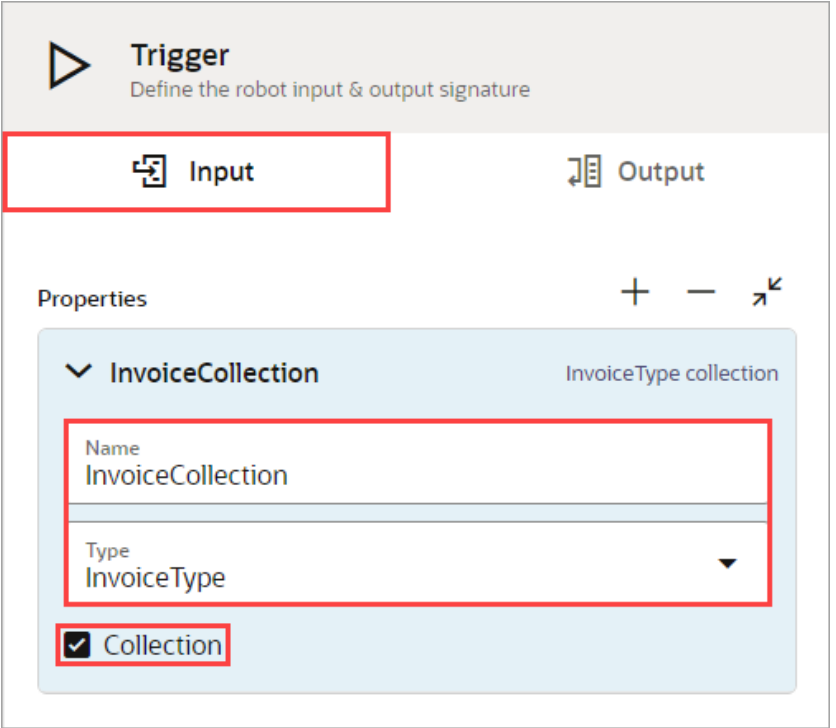
A robot completes the following tasks for a set of invoices:

- 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, InvoiceType, with the following properties:</p> <ul style="list-style-type: none"> • InvoiceNumber • InvoiceAmount • SupplierName • DueAmount 

See [Create a Data Type](#).

Define the trigger	<p>Create an input, InvoiceCollection, that is of the InvoiceType data type. Make the trigger a collection so that it can pass in an array of data—in this case, a number of invoices.</p> 
--------------------	---

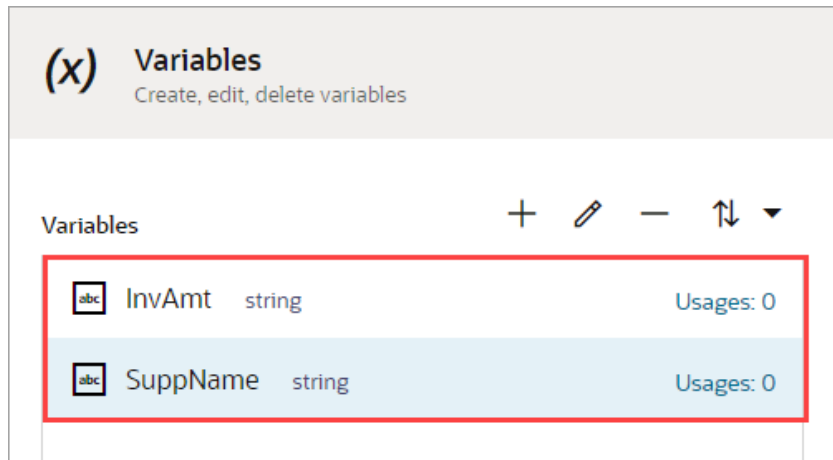
See [Create a Trigger's Input or Output](#).

Requirement	How to meet the requirement
-------------	-----------------------------

Define places to store invoice amount and supplier name data

Create the following variables:

- InvAmt
- SuppName

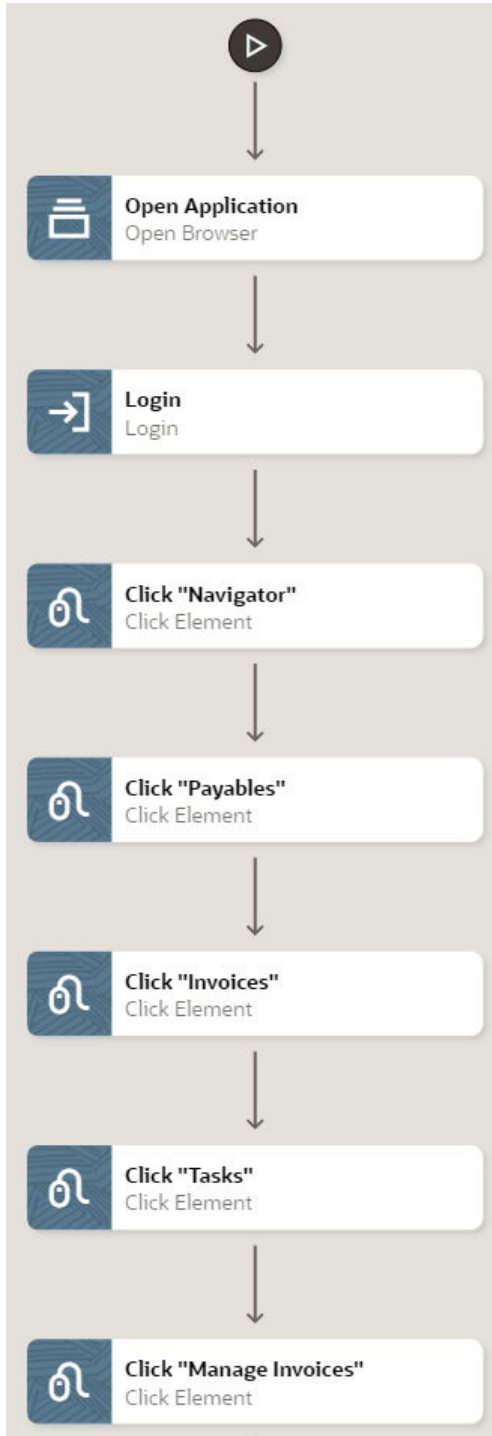


See [Create a Variable](#).

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:



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

Requirement	How to meet the requirement
-------------	-----------------------------

<p>Create a container for the actions that the robot must perform on every invoice</p>	<p>Add a foreach loop to the robot. The foreach loop defines how the robot iterates over the invoices.</p> <p>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.</p> <p>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.</p>
--	--

foreach
Execute actions for each element of a data set

Name*
foreachInvoice

Description

Collection*
\${INPUT.InvoiceCollection}

Iteration parameter*
CurrentInvoice

See [Add a Foreach Loop](#).

<p>Record the invoice number in the activity stream</p>	<p>Within the foreach loop, create a log action, which records the invoice number of every invoice that the robot updates. This information appears in the activity stream. Knowing the invoice number is helpful if you need to troubleshoot a robot instance.</p>
---	---

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]}

Variables
Drag and drop a variable or a property of the variable

Available variables

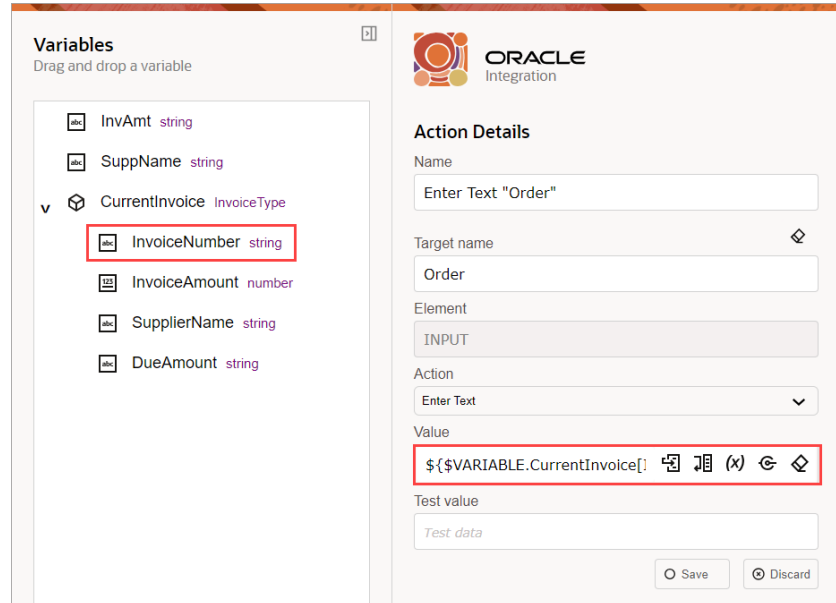
- InvAmt string
- SuppName string
- CurrentInvoice InvoiceType
 - InvoiceNumber string
 - InvoiceAmount number
 - SupplierName string
 - DueAmount string

See [Add a Log Action](#).

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.

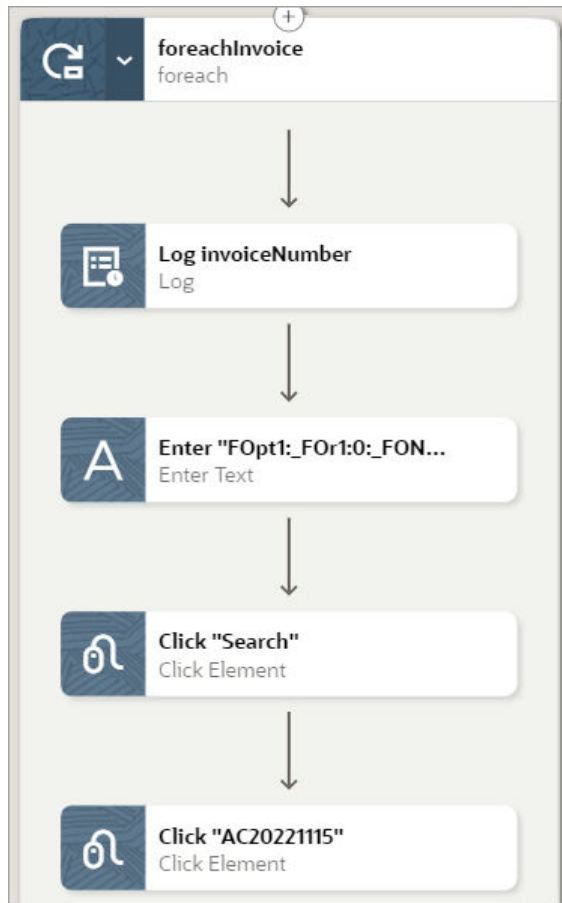


The screenshot displays the Oracle Integration Studio interface. On the left, the 'Variables' panel shows a tree view with 'CurrentInvoice' expanded, and 'InvoiceNumber' (string) highlighted with a red box. On the right, the 'Action Details' panel for an 'Enter Text' action is shown. The 'Name' field contains 'Enter Text "Order"'. The 'Target name' field contains 'Order'. The 'Element' field is set to 'INPUT'. The 'Action' dropdown is set to 'Enter Text'. The 'Value' field contains the expression `${$VARIABLE.CurrentInvoice[InvoiceNumber]}`, which is also highlighted with a red box. The 'Test value' field contains 'Test data'. At the bottom right, there are 'Save' and 'Discard' buttons.

See [Add an Enter Text Action](#).

Requirement	How to meet the requirement
-------------	-----------------------------

Open the invoice	Within the foreach loop, define additional required interactions to open an invoice.
------------------	--



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

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 configuration for a 'Get Text' action. In the 'Variables' panel, the 'SuppName' variable is highlighted. In the 'Action Details' panel, the 'Save to' field is set to `${VARIABLE.SuppName}`.

See [Add a Get Text Action](#).

Requirement**How to meet the requirement**

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 displays the Oracle Integration configuration interface. On the left, the 'Variables' panel lists several variables: 'InvAmt' (string), 'SuppName' (string), 'CurrentInvoice' (InvoiceType), 'InvoiceNumber' (string), 'InvoiceAmount' (number), 'SupplierName' (string), and 'DueAmount' (string). The 'InvAmt' variable is highlighted with a red box. On the right, the 'Action Details' panel shows the configuration for a 'Get Text' action. The 'Name' field is 'Get Text'. The 'Target name' field contains the XPath expression 'xpath_____id__pt1__FOR1_1__FONSr2_0_MAT2_0_'. The 'Element' field is 'TD'. The 'Action' dropdown is set to 'Get Text'. The 'Save to' field contains the expression '\${VARIABLE.InvAmt}', which is also highlighted with a red box. At the bottom right of the 'Action Details' panel are 'Save' and 'Discard' buttons.

See [Add a Get Text Action](#).

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:

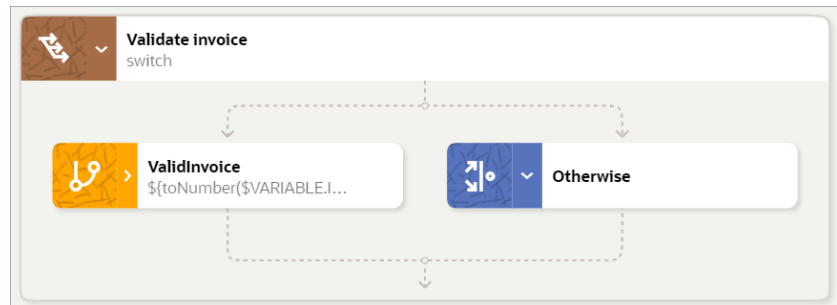
```


${toString($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.



See [Add a Switch Condition](#).

Use Case: Save Data After Iterating on Invoices

In this use case, an organization 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 identifies the invoice numbers to cancel by generating a report.

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. However, an integration can't perform this task. Here's why: The REST APIs for the application don't allow you to select 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

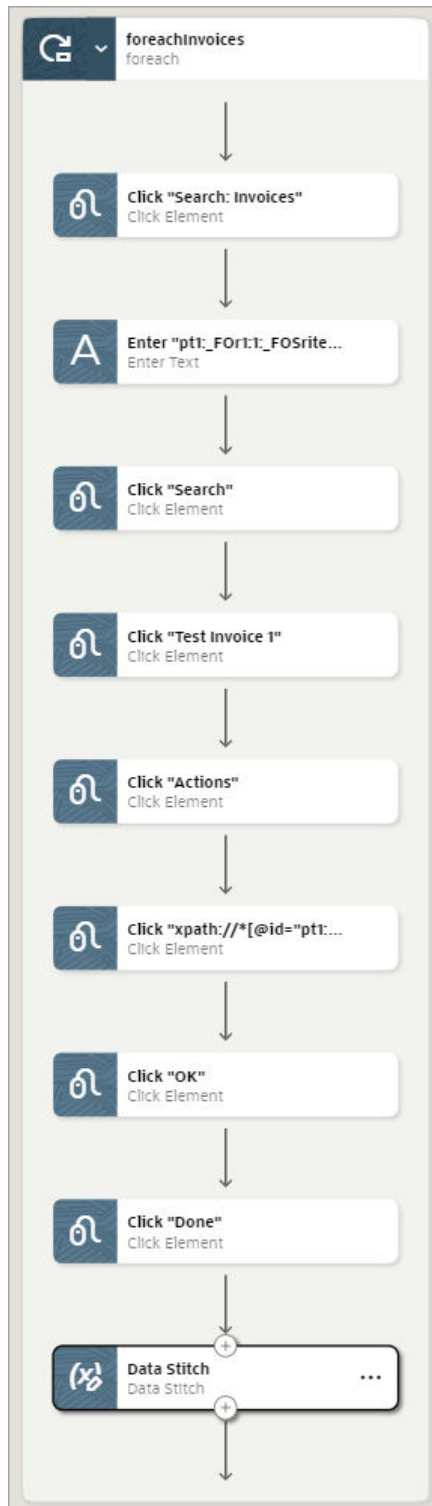
Requirement	How to meet the requirement
Create the data types for the trigger	<p>Define the data type for the input</p> <ul style="list-style-type: none"> • Name: InvoiceNumber • Property: Invoice <ul style="list-style-type: none"> – Type: string – Not a collection <p>Define the data type for the output</p> <ul style="list-style-type: none"> • Name: Result • Properties: <ul style="list-style-type: none"> – InvoiceNumber <ul style="list-style-type: none"> * Type: string * Not a collection – Status <ul style="list-style-type: none"> * Type: string * Not a collection <p>See Create a Data Type.</p>

Requirement	How to meet the requirement
Define the trigger	<p>Define the trigger's input</p> <ul style="list-style-type: none"> • Name: InvoiceNumber • Type: InvoiceNumber • Collection: Yes <p>The input allows the robot to receive a list of invoices from an integration. For example, the robot might receive the invoice numbers for 20 invoices.</p> <p>Define the trigger's output</p> <ul style="list-style-type: none"> • Name: Status • Type: Result • Collection: Yes <p>The output collects information for multiple invoices, so it must be a collection.</p> <p>The output allows the robot to collect the following information and pass it back to an integration:</p> <ul style="list-style-type: none"> • The invoice number of each canceled invoice. • The status of each canceled invoice. <p>See Create a Trigger's Input or Output.</p>
Define the variables	<p>Define a variable to hold the number of each updated invoice</p> <ul style="list-style-type: none"> • Name: CurrInvoice • Type: InvoiceNumber • Collection: No <p>Define a variable to hold the status of each updated invoice</p> <ul style="list-style-type: none"> • Name: CurrentInvoiceResult • Type: Result • Collection: No <p>See Create a Variable.</p>

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



See [Add a Foreach Loop](#) and [Add a Data Stitch Action](#).

Requirement

How to meet the requirement

Define the details of the data stitch

The data stitch contains the following operations:

(x) Data Stitch
Assign values to one or more variables

Name*
Data Stitch

Description
Stitch values to variables

↶ ↷ ↵ ↶

`$$$VARIABLE.CurrentInvoiceResult[InvoiceNumber]` ✕

= `$$$VARIABLE.CurrInvoice[Invoice]`

(x) Variable
`$$$VARIABLE.CurrentInvoiceResult[InvoiceNumber]`

= Operation
Assign

Value
`$$$VARIABLE.CurrInvoice[Invoice]`

`$$$VARIABLE.CurrentInvoiceResult[Status]` = ✕

`"Success"`

(x) Variable
`$$$VARIABLE.CurrentInvoiceResult[Status]`

= Operation
Assign

Value
`"Success"`

`$$$OUTPUT.Status` + ✕

`$$$VARIABLE.CurrentInvoiceResult`

(x) Collection
`$$$OUTPUT.Status`

+ Operation
Append

Value
`$$$VARIABLE.CurrentInvoiceResult`

- **First operation: Assign a value**
`$$$VARIABLE.CurrentInvoiceResult[InvoiceNumber]`

Requirement	How to meet the requirement
	<pre>= \${\$VARIABLE.CurrInvoice[Invoice]}</pre> <p>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.</p> <p>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.</p> <ul style="list-style-type: none"> Second operation: Assign a value <pre>`\${\$VARIABLE.CurrentInvoiceResult[Status]}</pre> <pre>= "Success"</pre> <p>This assignment assigns the Success value for each canceled invoice to the Status object in the CurrentInvoiceResult variable.</p> <p>Because each iteration of the foreach loop loads a new status, the Status object in the CurrentInvoiceResult variable holds each status only temporarily. Another operation in the data stitch records the value permanently.</p> Third operation: Append a value <pre>`\${\$OUTPUT.Status} +</pre> <pre>`\${\$VARIABLE.CurrentInvoiceResult}</pre> <p>This assignment appends the two objects in the CurrentInvoiceResult property, InvoiceNumber and Status, to the Status output. Because this operation is an append, the operation records all values for all invoices to the output.</p> <p>See Add a Data Stitch Action.</p>

Use Case: Save Values After Iterating on a Table

In this use case, an organization must save values from a table. Explore how the define web table action, data stitch action, and a foreach loop support this workflow.

Scenario

A robot completes the following tasks:

- Opens a dashboard, and views all invoices that have been placed on hold.
- Collects the following information for all on-hold invoices:
 - Invoice number
 - Hold reason
 - Invoice amount

Why Create a Robot?

An API is unable to capture the data from the table, or the team that is planning this automation work is unaware of the availability of an API that can do this work.

Workflow

Requirement	How to meet the requirement
Create a data type for the trigger's output	<ul style="list-style-type: none"> Name: Result Properties: <ul style="list-style-type: none"> InvoiceNumber <ul style="list-style-type: none"> Type: string Not a collection HoldReason <ul style="list-style-type: none"> Type: string Not a collection InvoiceAmount <ul style="list-style-type: none"> Type: string Not a collection

The screenshot displays two panels from the Oracle Integration Cloud interface. The left panel, titled 'Data type', shows the configuration for a data type named 'Result'. It includes a 'Name' field with the value 'Result', a 'Property list' button, and a 'Sample payload' button. Below these are the 'Properties' section, which lists three properties: 'InvoiceNumber' (string), 'HoldReason' (string), and 'InvoiceAmount' (string). The right panel, titled 'Data types', shows a list of existing data types: 'File' (Usages: 0), 'Define_Web_TableType' (Usages: 2), and 'Result' (Usages: 0). The 'Result' data type is highlighted with a red border.

See [Create a Data Type](#).

Requirement	How to meet the requirement
Define the trigger's output	<ul style="list-style-type: none"> Name: OnHoldInvoices Type: Result Collection: Yes

▶ **Trigger**
Define the robot input & output signature

🔌 Input
📄 Output

Properties + - ↗

▼ OnHoldInvoices
Result collection

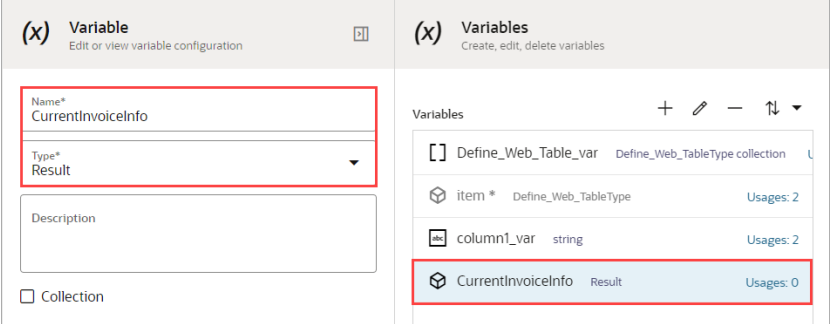
Name
OnHoldInvoices

Type
Result ▼

Collection

The output allows the robot to collect the invoice number, hold reason, and invoice amount for each on-hold invoice and pass this information back to the integration.

See [Create a Trigger's Input or Output](#).

Requirement	How to meet the requirement
Define the variables	<p>Define a variable to hold each invoice number, hold reason, and invoice amount</p> <ul style="list-style-type: none"> Name: <code>CurrentInvoiceInfo</code> Type: <code>Result</code> Collection: <code>No</code> <p>Because this variable is of the <code>Result</code> data type, it has three properties:</p> <ul style="list-style-type: none"> <code>InvoiceNumber</code> <code>HoldReason</code> <code>InvoiceAmount</code> <p>These properties hold the required data from the on-hold invoices. This variable collects the values obtained by the <code>get text</code> action. This variable also sends the values to the output using a <code>data stitch</code> action.</p>  <p>The screenshot shows two panels. The left panel, titled '(x) Variable', is for editing the configuration of the variable 'CurrentInvoiceInfo'. It shows the Name field set to 'CurrentInvoiceInfo', the Type dropdown set to 'Result', and a Description field. The 'Collection' checkbox is unchecked. The right panel, titled '(x) Variables', shows a list of variables. The variable 'CurrentInvoiceInfo' is highlighted with a red box. It is of type 'Result' and has 0 usages. Other variables in the list include 'Define_Web_Table_var' (collection type, 0 usages), 'item *' (Define_Web_TableType, 2 usages), and 'column1_var' (string, 2 usages).</p>

See [Create a Variable](#).


Requirement

How to meet the requirement

Identify the columns that contain the data that you need


Add a define web table action, and target the columns that contain the data that you need.


After you finish identifying the columns, the action looks similar to this:


 **Define Web Table**
Define an HTML table structure


Name*
Define Web Table

Description
Define an HTML table structure

 Input

 Output

 Pre Validate



>

Columns

+
-
↶

▼
Invoice number

Header*
\${\$TARGET.Define_Web_Table_Header_2}

Row 1*
\${\$TARGET.Define_Web_Table_Row_1_2}

Row 2*
\${\$TARGET.Define_Web_Table_Row_2}

▼
Hold reason

Header
\${\$TARGET.Define_Web_Table_Header_3}

Row 1
\${\$TARGET.Define_Web_Table_Row_1_3}

▼
Invoice amount

Header
\${\$TARGET.Define_Web_Table_Header_4}

Row 1
\${\$TARGET.Define_Web_Table_Row_1_4}

Cancel

OK

See [Add a Web Table Action](#).

Requirement	How to meet the requirement
-------------	-----------------------------

Add a foreach loop so that you can iterate over the XPath locators in the table

Add a foreach loop to the robot.

With the define web table action, you identify the XPath locators that you're interested in. The foreach loop allows you to iterate over these locators.

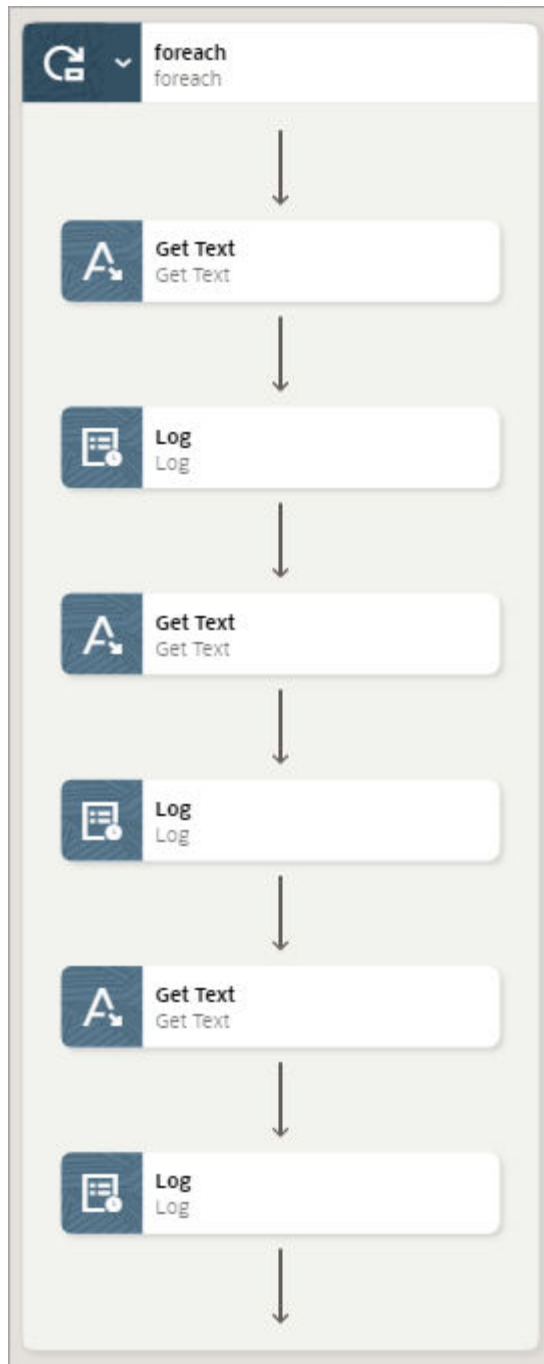
- For the **Collection** field, select the variable that the define web table action created for you.
- For the **Iteration parameter** field, enter a name for every value in the table, such as `item`.

(x) Variables	foreach
Drag and drop a variable or a property of the variable	Execute actions for each element of a data set
Available variables +	Name* foreach
[] Define_Web_Table_var Define_Web_TableType collec	Description
	Collection* \${VARIABLE.Define_Web_Table_var}
	Iteration parameter* item
	<input type="checkbox"/> Continue iteration on error

See [Add a Foreach Loop](#).

Requirement	How to meet the requirement
Get text from the table	<p data-bbox="643 247 748 273">Overview</p> <p data-bbox="643 283 1166 308">Within the foreach loop, add the following actions:</p> <ol data-bbox="643 327 1438 590" style="list-style-type: none"><li data-bbox="643 327 1398 352">1. Get text: Get the text from one cell in the <i>first column</i> you identified.<li data-bbox="643 373 1135 399">2. Log: Record the text to the activity stream.<li data-bbox="643 420 1438 445">3. Get text: Get the text from one cell in the <i>second column</i> you identified.<li data-bbox="643 466 1135 491">4. Log: Record the text to the activity stream.<li data-bbox="643 512 1406 537">5. Get text: Get the text from one cell in the <i>third column</i> you identified.<li data-bbox="643 558 1135 583">6. Log: Record the text to the activity stream. <p data-bbox="643 604 1438 630">The first iteration of the foreach loop gets text from the <i>first row</i> in the table.</p> <p data-bbox="643 636 1425 661">The second iteration gets text from the <i>second row</i> in the table, and so on.</p>

Requirement	How to meet the requirement
-------------	-----------------------------



How to define the get text actions

- On the Input tab for each get text action, for the **Locator** field, select the property for a column.

Requirement

How to meet the requirement

- On the Output tab, for the **Save to** field, drag the appropriate property for the `CurrentInvoiceInfo` variable. For example:
 - Use `InvoiceNumber` for the first get text action.
 - Use `HoldReason` for the second get text action.
 - Use `InvoiceAmount` for the third get text action.

How to define the log actions

For each log action, in the **Message** field, drag the appropriate property for the `CurrentInvoiceInfo` variable. For example:

- Use `InvoiceNumber` for the first log action.
- Use `HoldReason` for the second log action.
- Use `InvoiceAmount` for the third log action.

See [Add a Get Text Action](#) and [Add a Log Action](#).

Requirement	How to meet the requirement
Add a data stitch to the foreach loop, and define its details	<p>The data stitch doesn't contain an assign operation because the get text action already inserted values into the properties of the placeholder variable (<code>CurrentInvoiceInfo</code>). Therefore, the data stitch contains only the following an append operation. This operation appends the <code>CurrentInvoiceInfo</code> variable to the <code>OnHoldInvoices</code> output. Because this operation is an append, the operation records all values for all on-hold invoices to the output.</p> <pre> $\\${\\$OUTPUT.OnHoldInvoices} + \\${\\$VARIABLE.CurrentInvoiceInfo}$ </pre>

(x) **Data Stitch**
Assign values to one or more variables

Name*
 Data Stitch

Description
 Stitch values to variables

↗ ↖

$\${\$OUTPUT.OnHoldInvoices}$

+

$\${\$VARIABLE.CurrentInvoiceInfo}$

×

(x) Collection
 $\${\$OUTPUT.OnHoldInvoices}$

+

Operation
 Append

abc Value
 $\${\$VARIABLE.CurrentInvoiceInfo}$

+ Add assignment

See [Add a Data Stitch Action](#).

Use Case: Switch Browsers


In this use case, 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	How to meet the requirement
Add two open browser actions to the robot, as well as any additional actions	Build the robot as required, as shown in the following example. <div data-bbox="643 919 984 1808" style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;">  <p>The flowchart illustrates a sequence of actions in a robot workflow. It starts with a play button icon. The sequence consists of six action cards connected by downward arrows: 1. 'Open Application' (Open Browser) - highlighted with a red box. 2. 'Get Text' (Get Text). 3. 'Log' (Log). 4. 'Open Browser' (Open Browser) - highlighted with a red box. 5. 'Get Text' (Get Text). 6. 'Log' (Log).</p> </div>
	See Add an Action to a Robot .

Requirement

How to meet the requirement

Use a variable to define the output for each open browser action

For example, create two variables, `index1` and `index2`, to store the index value for each open browser action.

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.

See [Create a Variable](#).

When the robot needs to switch back to the first internet browser, add a switch browser action to the robot

Define the **Browser window** field using the `index1` 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.

See [Add a Switch Browser Action](#).

Use Case: Switch Windows

In this use case, a robot must complete work in several pop-up windows, including a pop-up window that is launched from another pop-up window. Explore how the window action supports this workflow.

Scenario

A robot completes the following tasks:

1. Opens a web application and completes a task that opens a pop-up window (child window 1).
2. Completes a task in child window 1. The task opens another pop-up window (child window 2).
3. Closes the two child windows, and returns to the main window.

Workflow

This workflow focuses on the steps that require the window action.

Task	More information
Create variables for storing the titles of the windows that the robot interacts with	<p>For example, create the following string variables to store the handles for the windows that the robot interacts with:</p> <ul style="list-style-type: none"> • mainWindowHandle • childWindowHandle1 • childWindowHandle2 <p>Additionally, create the following string collection variables to hold all window handles and titles:</p> <ul style="list-style-type: none"> • windowHandles • windowTitles <p>See Create a Variable.</p>
Add actions that open the web application and open child window 1	<p>Add robot actions as needed.</p> <p>See Add an Action to a Robot.</p>

Task

Add a window action that switches the robot's focus to child window 1

More information

The following action switches to child window 1 by specifying **NEW** for the **Window Id**.

Window
Interact with browser windows

Name*
Switch to child window 1

Description
Interact with browser windows

Operations
Switch To

Input **Output** **Post Validate**


Window Id*
NEW

Timeout (seconds)
5

Full page screenshot after this action

On the Output tab, the handle for the main window is saved to a variable. The variable lets the robot easily return to the main window later in the flow. See [Add a Window Action](#).




Task	More information
------	------------------


Window
Interact with browser windows

Name*
Switch to child window 1

Description
Interact with browser windows

Operations
Switch To ▼

 Input
 Output
 Post Validate

Save to
\${\$VARIABLE.mainWindowHandle}

Interact with child window 1, prompting child window 2 to open


Add robot actions as needed.
See [Add an Action to a Robot](#).

Task

Add a window action that switches the robot's focus from child window 1 to child window 2

More information

The following action switches to child window 2 by specifying **NEW** for the **Window Id**.



Window

Interact with browser windows

Name*


Switch to child window 2


Description


Interact with browser windows

Operations

Switch To ▼


Input


Output


Post Validate

Window Id*





NEW

Timeout (seconds)

5 ▼ ▲

Full page screenshot after this action

On the Output tab, the handle for child window 1 is saved to a variable. The variable lets the robot easily return to the window later in the flow. See [Add a Window Action](#).


Task	More information
	<div data-bbox="646 247 1442 982" style="border: 1px solid #ccc; padding: 10px;"> <div style="background-color: #f0f0f0; padding: 5px; border-bottom: 1px solid #ccc;">  Window Interact with browser windows </div> <div style="padding: 5px; border: 1px solid #ccc; margin-top: 5px;"> Name* Switch to child window 2 </div> <div style="padding: 5px; border: 1px solid #ccc; margin-top: 5px;"> Description Interact with browser windows </div> <div style="padding: 5px; border: 1px solid #ccc; margin-top: 5px;"> Operations Switch To ▼ </div> <div style="display: flex; justify-content: space-around; margin-top: 10px;">  Input  <u>Output</u>  Post Validate </div> <div style="padding: 5px; border: 1px solid #ccc; margin-top: 10px;"> Save to \${\$VARIABLE.childWindowHandle1} </div> </div>
Interact with child window 2	Add robot actions as needed. See Add an Action to a Robot .

Task

Get and log the window handles and titles of all the windows that the robot has interacted with up until now

More information


Add two window actions. The first action gets the handles and saves them in a string collection variable. The second action gets the titles and saves them in a string collection variable. See [Add a Window Action](#).

 **Window**
Interact with browser windows


Name*
Get window handles

Description
Interact with browser windows

Operations
Get Window Handles ▼

 **Output**


Save to
\${\$VARIABLE.windowHandles}

 **Window**
Interact with browser windows

Name*
Get window titles

Description
Interact with browser windows

Operations
Get Window Titles ▼

 **Output**


Save to
\${\$VARIABLE.windowTitles}

Additionally, add two log actions to log the values to the activity stream. See [Add a Log Action](#).

Task **More information**

Close child window 2, and switch the focus to child window 1

Add two window actions. The first action closes child window 2. See [Add a Window Action](#).


 **Window**
Interact with browser windows

Name*
Close child window 2

Description
Interact with browser windows

Operations
Close Window ▼

The second action switches focus to child window 1 by specifying the variable for this window.

 **Window**
Interact with browser windows

Name*
Switch to child window 1

Description
Interact with browser windows

Operations
Switch To ▼

Input
 Output
 Post Validate

Window Id*
\${\$VARIABLE.childWindowHandle1}


Timeout (seconds)
5 ▼ ▲

Full page screenshot after this action

Task **More information**

Close child window 1, and switch the focus to the main window

Add two window actions. The first action closes child window 1. See [Add a Window Action](#).

 **Window**
Interact with browser windows

Name*

Close child window window 1


Description

Interact with browser windows

Operations

Close Window ▼

The second action switches focus to the main window by specifying its variable.

 **Window**
Interact with browser windows

Name*




Switch to main window

Description

Interact with browser windows

Operations

Switch To ▼

 **Input**  **Output**  **Post Validate**

Window Id*

`${$VARIABLE.mainWindowHandle}`

Timeout (seconds)

5 ▼ ▲

Full page screenshot after this action